

# A comparative study of iterative solvers exploiting spectral information for SPD systems

L. Giraud\*      D. Ruiz†      A. Touhami†‡

CERFACS Technical Report TR/PA/04/40  
Also ENSEEIHT-IRIT Technical Report RT/TLSE/04/03

## Abstract

When solving the Symmetric Positive Definite (SPD) linear system  $\mathbf{Ax} = \mathbf{b}$  with the conjugate gradient method, the smallest eigenvalues in the matrix  $\mathbf{A}$  often slow down the convergence. Consequently if the smallest eigenvalues in  $\mathbf{A}$  could be somehow “removed”, the convergence may be improved. This observation is of importance even when a preconditioner is used, and some extra techniques might be investigated to further improve the convergence rate of the conjugate gradient on the given preconditioned system. Several techniques have been proposed in the literature that either consist of updating the preconditioner or enforcing conjugate gradient to work in the orthogonal complement of an invariant subspace associated with smallest eigenvalues. Among these approaches we consider first a two-phase algorithm using a deflation-type idea. In a first stage this algorithm computes a partial spectral decomposition simply using matrix-vector products. More precisely it combines Chebyshev iterations with a block Lanczos procedure to accurately compute an orthogonal basis of the invariant subspace associated with the smallest eigenvalues. Then, the solution on this subspace is computed using a projector while the solution in the orthogonal complement is obtained with Chebyshev iterations that benefit from the reduced condition number.

For sake of comparison, this eigen-information is used in combination with other techniques. In particular we consider the deflated version of conjugate gradient. As representative of techniques exploiting the spectral information to update the preconditioner we consider also the approaches that attempt to shift the smallest eigenvalues close to one where most of the eigenvalues of the preconditioned matrix should be located. Finally we consider an algebraic two-grid scheme inspired by ideas from the multigrid philosophy. In this paper, we describe these various variants as well as the observed numerical behavior on a set of model problems from Matrix Market or arising from the discretization via finite element technique of some 2D heterogeneous diffusion PDE problems. We discuss their numerical efficiency, computational complexity and sensitivity to the accuracy of the eigen-calculation.

**Keywords.** Chebyshev polynomials, block Lanczos method, conjugate gradient method, filtering, deflation, spectral preconditioning, two-grid schemes, iterative methods.

---

\*CERFACS, 42 Avenue G. Coriolis, 31057 Toulouse Cedex, France

†ENSEEIHT-IRIT, 2 rue C. Camichel, 31071 Toulouse Cedex, France

‡Correspondence to: Ahmed.Touhami@enseeiht.fr

# 1 Introduction

The Krylov solver of choice for the solution of large sparse and symmetric positive linear systems of the form

$$\mathbf{Ax} = \mathbf{b} \quad (1)$$

is the conjugate gradient [13], which constructs the  $k$ th iterate  $\mathbf{x}^{(k)}$  as an element of

$$\mathbf{x}^{(0)} + \text{Span}\{\mathbf{r}^{(0)}, \dots, \mathbf{A}^{(k-1)}\mathbf{r}^{(0)}\}$$

so that  $\|\mathbf{x}^{(k)} - \mathbf{x}^*\|_{\mathbf{A}}$  is minimized, where  $\mathbf{x}^*$  is the exact solution of (1). The convergence of conjugate gradient is governed by the eigenvalues distribution in  $\mathbf{A}$  [27]. A coarse characterization is given by the bound on the rate of convergence of the conjugate gradient method given by Concus et al. [5] viz

$$\|\mathbf{e}^{(k)}\|_{\mathbf{A}} \leq 2 \|\mathbf{e}^{(0)}\|_{\mathbf{A}} \left( \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^k, \quad (2)$$

where  $\mathbf{e}^{(k)} = \mathbf{x}^* - \mathbf{x}^{(k)}$  is the error associated with the iterate at step  $k$ ,  $\kappa = \lambda_{max}/\lambda_{min}$  denotes the condition number of  $\mathbf{A}$ , and the  $\mathbf{A}$ -norm of  $\mathbf{x}$  is given by  $\|\mathbf{x}\|_{\mathbf{A}} = (\mathbf{x}^T \mathbf{Ax})^{1/2}$ . From this bound, it can be seen that increasing the size of the smallest eigenvalues will decrease substantially the upper bound on the convergence rate of the conjugate gradient. Clearly the presence of eigenvalues close to the origin makes this difficult. Thus removing the bad effect of small eigenvalues in the matrix  $\mathbf{A}$  might have a beneficial effect on the convergence. Several techniques have been proposed in the literature that either consists of updating the preconditioner or enforcing conjugate gradient to work in the orthogonal complement of an invariant subspace associated with small eigenvalues.

The aim of this work is to compare several of these techniques in terms of numerical efficiency. Among these techniques we consider first the two-phase approach recently proposed in [2]. Analogously to direct methods, it includes a preliminary phase which we call “partial spectral factorization phase” followed by a solution phase, both only based on numerical tools that are usually exploited in iterative methods.

In a first stage, this algorithm computes (with a predetermined accuracy) a basis for the invariant subspace associated with the smallest eigenvalues in  $\mathbf{A}$ . It uses Chebyshev polynomials as a spectral filtering tool in  $\mathbf{A}$ , to damp in a set of vectors, eigencomponents associated with the largest eigenvalues in  $\mathbf{A}$  and maintain, in the sequence of Krylov vectors, eigencomponents associated with the largest eigenvalues in  $\mathbf{A}$  under a given level of filtering. This phase is referred to as “partial spectral factorization”. Chebyshev iteration can also be used to compute the part of the solution associated with the largest eigenvalues in  $\mathbf{A}$ . The second part of the solution, associated with the smallest eigenvalues in  $\mathbf{A}$ , is obtained with an oblique projection of the initial residual onto the “near”-invariant subspace, in order to get those eigencomponents in the solution corresponding to the smallest eigenvalues in  $\mathbf{A}$ , and exploits the precomputed partial spectral factorization of  $\mathbf{A}$ .

In this paper, the computation of an orthonormal basis  $\mathbf{W}$  using the “partial spectral factorization” is exploited in combination with other approaches. In particular we

consider the deflated version of conjugate gradient described in [25]. This algorithm exploits the link between the Lanczos algorithm and the standard conjugate gradient algorithm, and is mathematically equivalent to the Nicolaide’s algorithm which is derived from a “deflated” Lanczos procedure [18]. In [15] Kolotilina uses a twofold deflation technique for simultaneously deflating the  $k$  largest and the  $k$  smallest eigenvalues using an appropriate deflating subspace of dimension  $k$ . An alternative technique consists in applying the deflation to eliminate the bad effects on the convergence caused by the remaining small eigenvalues [29]. This technique is used in [6] to solve electromagnetic problems with large jumps in the coefficients. As representative of techniques exploiting the spectral information to update the preconditioner we consider the approach proposed in [3] that attempts to shift the smallest eigenvalues close to one where most of the eigenvalues of the preconditioned matrix should be located. Finally we consider an algebraic two-grid scheme inspired by ideas from the multigrid philosophy.

The outline of the paper is as follows. In Section 2, we describe the Chebyshev-based partial spectral factorization. We devote Section 3 to present various solution techniques that exploit the spectral information about the matrix  $\mathbf{A}$ , and indicate how they can be combined with “partial spectral factorization” to construct efficient iterative solvers. In Section 4, we compare these techniques in terms of numerical efficiency and computational complexity on a set of model problems from Matrix Market, or arising from the discretization via finite element technique of some heterogeneous diffusion PDEs in 2D. Finally, we conclude with some open questions and remarks about the cost benefit of this approach in Section 5.

## 2 A Chebyshev-based partial spectral decomposition

In this section, we introduce a Chebyshev-based filtering technique combined with some Lanczos process that is used to compute a basis for the invariant subspace associated with the smallest eigenvalues in  $\mathbf{A}$ . To do so, we start with an initial set of  $s$  randomly generated vectors ( $s$  is the block-size) which we orthonormalize, and we use Chebyshev polynomials in  $\mathbf{A}$  to “*damp*”, in these starting vectors, the eigenfrequencies associated with all the eigenvalues in some predetermined range. We can fix a positive number  $\lambda_{\min}(\mathbf{A}) < \mu < \lambda_{\max}(\mathbf{A})$ , and decide to compute all the eigenvectors associated with all the eigenvalues in the range  $[\lambda_{\min}(\mathbf{A}), \mu]$ . The computation of  $\lambda_{\max}(\mathbf{A})$  is usually not too difficult, and in some cases, a sharp upper-bound may be already available through some *a priori* knowledge of the numerical properties of  $\mathbf{A}$ . If the eigenvalues of  $\mathbf{A}$  are well clustered, the number of remaining eigenvalues inside the predetermined range  $[\lambda_{\min}(\mathbf{A}), \mu]$ , (with reasonable  $\mu$  like  $\lambda_{\max}/100$ , or  $\lambda_{\max}/10$ , for instance) should be small compared to the size of the linear system. Then, after “*filtering*” the initial starting vectors, we obtain a set of  $s$  vectors with eigencomponents filtered close to 0 for those eigenvalues in the range  $[\mu, \lambda_{\max}(\mathbf{A})]$ , and relatively much bigger eigencomponents linked with the smallest eigenvalues in  $\mathbf{A}$ . We then use a Lanczos/Block-Lanczos type of approach (see [9, 19, 26] for details on such techniques) to build a “*near*”-invariant basis starting with these filtered vectors.

In the next paragraph, we detail how the Chebyshev polynomials are incorporated, and in paragraph 2.2, we detail the combination we propose to build a “*near*”-invariant

subspace associated with the smallest eigenvalues in  $\mathbf{A}$ .

## 2.1 The Chebyshev based filtering technique

Let us denote by

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T = \mathbf{U}_1\mathbf{\Lambda}_1\mathbf{U}_1^T + \mathbf{U}_2\mathbf{\Lambda}_2\mathbf{U}_2^T.$$

the eigendecomposition of the SPD matrix  $\mathbf{A}$ ,  $\mathbf{\Lambda}_1$  being the diagonal matrix made with all eigenvalues of  $\mathbf{A}$  less than  $\mu$  and  $\mathbf{U}_1$  the unitary matrix whose columns are the corresponding orthonormalized eigenvectors in matrix form, and  $\mathbf{\Lambda}_2$  and  $\mathbf{U}_2$  the complementary corresponding matrices. Let  $\mathbf{P}$  a set of  $s$  randomly generated and orthonormalized vectors and  $\mathcal{F}_m$  a polynomial of degree  $m$  such that

$$\mathcal{F}_m(\mathbf{A})\mathbf{P} = \mathbf{U}_1\mathcal{F}_m(\mathbf{\Lambda}_1)\mathbf{U}_1^T\mathbf{P} + \mathbf{U}_2\mathcal{F}_m(\mathbf{\Lambda}_2)\mathbf{U}_2^T\mathbf{P} \quad (3)$$

Our aim is to choose a polynomial  $\mathcal{F}_m$  such that  $\mathcal{F}_m(\mathbf{\Lambda}_2)$  is close enough to 0 in some controlled manner, in order to get  $\mathcal{F}_m(\mathbf{A})\mathbf{P}$  very colinear to  $\mathbf{U}_1$ .

A family of polynomials which does this work are the Chebyshev polynomials which can be defined by the following 2-term recurrence formula (see [12, page 46]):

$$\begin{cases} T_0(\omega) = 1, & T_1(\omega) = \omega, \\ T_{m+1}(\omega) = 2\omega T_m(\omega) - T_{m-1}(\omega) & m \geq 1. \end{cases} \quad (4)$$

The optimal properties of Chebyshev polynomials given in Theorem 4.2.1 in [12, page 47] can be summarized as follows: if we consider  $|d| > 1$  and

$$H_m(\omega) = \frac{T_m(\omega)}{T_m(d)},$$

then  $H_m$  has minimum  $l_\infty$  norm on the interval  $[-1, 1]$  over all polynomials  $Q_m$  of degree less than or equal to  $m$  and satisfying the condition  $Q_m(d) = 1$ , and we have

$$\max_{\omega \in [-1, 1]} |H_m(\omega)| = \frac{1}{|T_m(d)|}. \quad (5)$$

Now, consider the translation plus homothetic transformation:

$$\lambda \in \mathbb{R} \mapsto \omega_\mu(\lambda) = \frac{\lambda_{\max}(\mathbf{A}) + \mu - 2\lambda}{\lambda_{\max}(\mathbf{A}) - \mu},$$

with  $\lambda_{\min}(\mathbf{A}) < \mu < \lambda_{\max}(\mathbf{A})$  given above. This transformation maps  $\lambda_{\max}(\mathbf{A})$  to  $-1$ ,  $\mu$  to  $1$ , and  $0$  to

$$\omega_\mu(0) = d_\mu = \frac{\lambda_{\max}(\mathbf{A}) + \mu}{\lambda_{\max}(\mathbf{A}) - \mu} > 1.$$

Then, introducing

$$\mathcal{F}_m(\lambda) = \frac{T_m(\omega_\mu(\lambda))}{T_m(d_\mu)}, \quad (6)$$

we easily see, because of the optimal properties (5), that  $\mathcal{F}_m(\lambda)$  has minimum  $l_\infty$  norm on the interval  $[\mu, \lambda_{\max}(\mathbf{A})]$  over all polynomials  $Q_m$  of degree less than or equal to  $m$

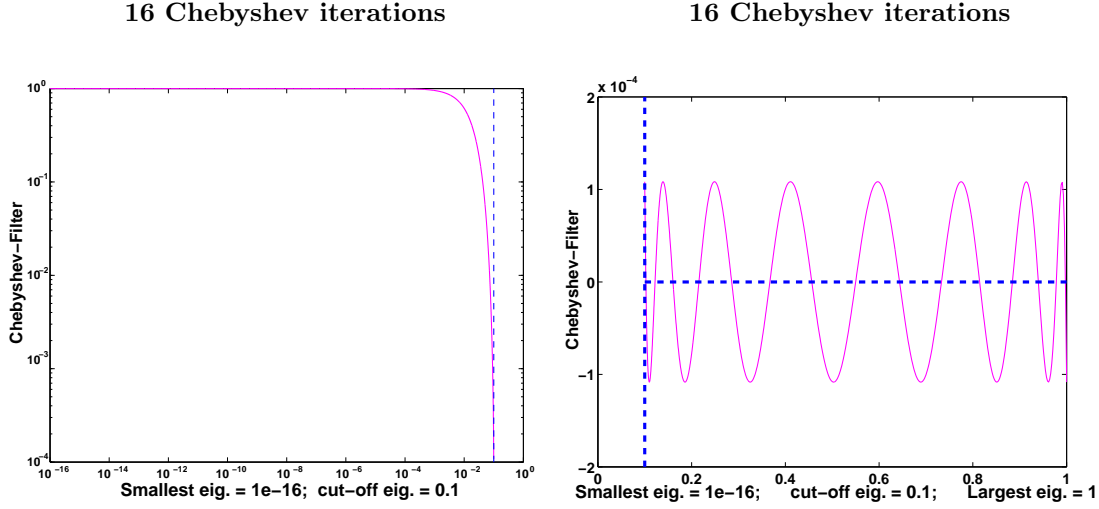


Figure 1: Use of Chebyshev polynomials in the range  $[\mu, \lambda_{max}]$ , with  $\mu = \lambda_{max}/10$ .

satisfying  $Q_m(0) = 1$ .

The example in Figure 1 shows the values of the Chebyshev polynomial  $\mathcal{F}_{16}(x)$ ,  $x \in ]0, 1]$ , as defined in (6) with  $\lambda_{min} = 10^{-16}$ ,  $\lambda_{max} = 1$ , and  $\mu = 10^{-1}$ . On the right side of Figure 1, we plot the value of  $\mathcal{F}_{16}(x)$  on the range  $[\mu, \lambda_{max}]$ , to illustrate how we can manage, with Chebyshev polynomials, a spectral filter on the interval  $[\lambda_{max}/10, \lambda_{max}]$  bounded uniformly below  $\varepsilon = 10^{-4}$ . On the left side, we plot in Log-Log Scale  $\mathcal{F}_{16}(x)$  on the interval  $[\lambda_{min}, \mu]$ . We can see that the fixed value  $\mathcal{F}_{16}(0) = 1$  maintains, because of continuity, the values of  $\mathcal{F}_{16}(\lambda)$  close to 1 for all  $\lambda \leq 10^{-2}$ .

Let us denote by

$$\mathbf{Z} = \text{Chebyshev-Filter}(\mathbf{P}, \varepsilon, [\mu, \lambda_{max}], \mathbf{A})$$

the application of a Chebyshev polynomial in  $\mathbf{A}$  to a set of vectors  $\mathbf{P}$ , viz.

$$\mathbf{Z} = \mathcal{F}_m(\mathbf{A}) \mathbf{P},$$

where  $\mathcal{F}_m$  is defined as in (6), with a degree such that its  $l_\infty$  norm over the interval  $[\mu, \lambda_{max}]$  is less than  $\varepsilon$ , with  $\varepsilon \ll 1$  being an *a priori* chosen level of filtering.

For given values of  $\mu$ ,  $\lambda_{max}(\mathbf{A})$  and  $\varepsilon$ , we can fix the degree  $m$  of  $T_m$  such that  $1/|T_m(d_\mu)| < \varepsilon$  on  $[\mu, \lambda_{max}(\mathbf{A})]$ , out of which we obtain  $\|\mathcal{F}_m(\mathbf{A}_2)\|_\infty < \varepsilon$ , and using (3) we can write

$$\|\mathbf{U}_2^T \mathbf{Z}\|_2 \leq \|\mathcal{F}_m(\mathbf{A}_2)\|_2 \|\mathbf{U}_2^T \mathbf{P}\|_2 \leq \varepsilon \|\mathbf{U}_2^T \mathbf{P}\|_2. \quad (7)$$

Equation (7) explains explicitly how the action of the Chebyshev filtering in  $\mathbf{A}$  applied on a set of vectors can reduce below a value  $\varepsilon$  the eigencomponents, with respect to the invariant subspace  $\mathbf{U}_2$  linked to all eigenvalues in the range  $[\mu, \lambda_{max}(\mathbf{A})]$ , relatively to

the others. The number of Chebyshev iterations to achieve a given level of filtering  $\varepsilon$ , is directly related to the rate of convergence of Chebyshev polynomials on the interval  $[\mu, \lambda_{max}]$  (see, e.g., [12, 8] which depends only on the ratio  $\lambda_{max}/\mu$ . As illustrated in Figure 1, 16 Chebyshev iterations are enough to reach a level  $\varepsilon = 10^{-4}$  on the interval  $[\mu, \lambda_{max}]$  with  $\lambda_{max}/\mu = 10$ . If we take  $\lambda_{max}/\mu = 100$ , for instance the number of Chebyshev iterations to reach the same level will then become 50.

## 2.2 The partial spectral factorization

The purpose of this paragraph is to detail partly the combination of Chebyshev filtering polynomials, as introduced in the paragraph 2.1, with Lanczos/Block-Lanczos type techniques [20, 22] to derive a “near”-invariant subspace of  $\mathbf{A}$  associated with the smallest eigenvalues in  $\mathbf{A}$ . The technique we propose, which corresponds to CHEBYSHEV-PSF above, can be summarized as follows.

We first need some information on the value of the largest eigenvalue in  $\mathbf{A}$ . This can be obtained, for instance with some steps of the power method (see [10, 24]). But we may also input some “not too large” upper bound on  $\lambda_{max}(\mathbf{A})$  that can be already available due to some properties of the given matrix  $\mathbf{A}$ . To define the filtering interval  $[\mu, \lambda_{max}(\mathbf{A})]$  that will be incorporated in the Chebyshev polynomials as explained in the previous § 2.1, we then fix the cut-off eigenvalue  $\mu$  to  $\lambda_{max}(\mathbf{A})/10$  or  $\lambda_{max}(\mathbf{A})/100$ , to control explicitly the rate of convergence of the Chebyshev polynomials  $\mathcal{F}_m(x)$  on the interval  $[\mu, \lambda_{max}(\mathbf{A})]$ , with the constraint that  $\mathcal{F}_m(0) = 1$  (see § 2.1). The value of  $\mu$ , as we will see in the following, also controls the reduced condition number  $\lambda_{max}(\mathbf{A})/\mu$  that we will implicitly exploit after derivation of a “near”-invariant subspace associated with all eigenvalues in  $\mathbf{A}$  in the range  $]0, \mu]$ . We can highlight already that there is a compromise to reach in this choice of  $\mu$ , since larger values of  $\mu$  may imply more eigenvalues (and associated eigenvectors) in the range  $]0, \mu]$ , and smaller values of  $\mu$  will increase the number of Chebyshev iterations needed to reach some predetermined filtering level  $\varepsilon$  on the interval  $[\mu, \lambda_{max}(\mathbf{A})]$ .

The purpose of step 3 to 5 in CHEBYSHEV-PSF are then to perform one initial step of Chebyshev filtering polynomials in  $\mathbf{A}$  to generate an initial orthonormal set  $\mathbf{W}^{(0)}$  of  $s$  vectors, (with  $s \geq 1$  being some given block size as in Block-Lanczos techniques) with the property that the eigencomponents in  $\mathbf{W}^{(0)}$  corresponding to eigenvalues in  $\mathbf{A}$  in the range  $[\mu, \lambda_{max}(\mathbf{A})]$  are (in absolute value) below some fixed predetermined value  $\varepsilon \ll 1$  (denoted as “filtering level” in § 2.1). With the availability of these pre-filtered orthonormal set of vectors, we can then start a Block-Lanczos iteration, corresponding to steps i to ii in CHEBYSHEV-PSF, to compute a “near”-invariant subspace associated with all eigenvalues in  $\mathbf{A}$  in the range  $]0, \mu[$ .

Now, the Lanczos/Block-Lanczos process is to build

$$\mathbf{P}^{(k+1)} = (\mathbf{I} - \mathbf{W}\mathbf{W}^T)\mathbf{A}\mathbf{W}^{(k)}$$

and to orthonormalize the set of  $s$  vectors  $\mathbf{P}^{(k+1)}$  to get the next entry  $\mathbf{W}^{(k+1)}$  in the block Krylov orthonormal basis  $\mathbf{W}$ . As discussed in [2], the major trouble of the Lanczos/Block-Lanczos iterations is that it deteriorates gradually the gap between the eigencomponents in the Lanczos vectors  $\mathbf{W}$ , initially set to about  $\varepsilon$ .

**ALGORITHM 1: CHEBYSHEV-PSF****Input:**  $\mathbf{A} \in \mathbb{R}^{n \times n}$ **Output:** An orthonormal basis  $\mathbf{W}$  associated with the smallest eigenvalues of  $\mathbf{A}$ .**Begin****Computation of  $\lambda_{max}(\mathbf{A})$  and choice of  $\mu$** 

1. Compute  $\lambda_{max}(\mathbf{A})$  using power method
2.  $\mu = \lambda_{max}(\mathbf{A})/\alpha$ , ( $\alpha = 10$ , or 100 for instance)

**Filtering of the initial generated basis**

3.  $\mathbf{P}^{(0)} = \text{random}(n, s)$
4.  $\mathbf{Z}^{(0)} = \text{Chebyshev-Filter}(\mathbf{P}^{(0)}, \varepsilon, [\mu, \lambda_{max}], \mathbf{A})$
5.  $\mathbf{W}^{(0)} = \text{orthonormalize}(\mathbf{Z}^{(0)})$
6.  $\mathbf{W} = \mathbf{W}^{(0)}$ ; and  $\delta_2 = 1$
7. **For**  $k = 0, 1, \dots$ , until convergence **Do:**

**Lanczos/Block-Lanczos Process**

- i.  $\mathbf{P}^{(k+1)} = (\mathbf{I} - \mathbf{W}\mathbf{W}^T)\mathbf{A}\mathbf{W}^{(k)}$
- ii.  $[\mathbf{Q}^{(k+1)}, \boldsymbol{\Sigma}_1^{(k+1)}, \mathbf{V}] = \text{SVD}(\mathbf{P}^{(k+1)}, 0)$
- iii.  $\delta_1 = \min(\text{diag}(\boldsymbol{\Sigma}_1^{(k+1)}))$

**Maintain the level of filtering**

- iv.  $\delta = \max(\varepsilon, \delta_1 \times \delta_2)$
- v.  $\mathbf{Z}^{(k+1)} = \text{Chebyshev-Filter}(\mathbf{Q}^{(k+1)}, \delta, [\mu, \lambda_{max}], \mathbf{A})$
- vi.  $\mathbf{Y}^{(k+1)} = (\mathbf{I} - \mathbf{W}\mathbf{W}^T)\mathbf{Z}^{(k+1)}$
- vii.  $[\mathbf{W}^{(k+1)}, \boldsymbol{\Sigma}_2^{(k+1)}, \mathbf{V}] = \text{SVD}(\mathbf{Y}^{(k+1)}, 0)$
- viii.  $\delta_2 = \min(\text{diag}(\boldsymbol{\Sigma}_2^{(k+1)}))$

**Incorporate the vectors in the current basis**

- ix.  $\mathbf{W} = [\mathbf{W}; \mathbf{W}^{(k+1)}]$
8. **EndDo**

**End**

SVD( $\mathbf{Z}, 0$ ) denotes the economy Singular Value Decomposition of  $\mathbf{Z}$  as in MATLAB notations.

This deterioration can be directly estimated (in terms of upper bounds) with the computation of the minimum singular value  $\delta_1$  (as in step iii) when orthonormalizing

the current set of Krylov vectors  $\mathbf{P}^{(k+1)}$ . Then, the purpose of the following steps (iv to v) is to maintain under the level  $\varepsilon$ , with a few extra Chebyshev filtering iterations, the relative gap in the eigencomponents associated with all eigenvalues in the range  $[\mu, \lambda_{max}(\mathbf{A})]$  in the newly computed Block-Lanczos vectors  $\mathbf{W}^{(k+1)}$ .

Finally, steps vi to viii, simply aim to orthogonalize the filtered vectors against the current orthonormal basis  $\mathbf{W}$  and to build an orthonormal set of new Lanczos vectors. We mention that it is necessary to perform a full reorthogonalization since the application of the polynomial filter in  $\mathbf{A}$  to the current set of vectors destroys the nice embedded property of Krylov subspaces with respect to the powers in  $\mathbf{A}$ .

The way of testing convergence or “near”-invariance with respect to  $\mathbf{U}_1$  can be achieved with the generation an extra random vector at the beginning, to filter it under the level  $\varepsilon$  and normalize it, and to test when appropriate (e.g. when  $\delta_2$  becomes small) if the norm of the orthogonal projection of this vector onto the orthogonal complement of the computed basis  $\mathbf{W}$  is close to  $\varepsilon$  or not. A priori convergence should be reached exactly when the size of the current basis  $\mathbf{W}$  is very close to the actual number of eigenvalues in  $\mathbf{A}$  below the value of  $\mu$ .

### 2.3 Some remarks

As explained by equation (7), the use of Chebyshev filtering polynomials in the context of the Lanczos/Block-Lanczos iteration can be viewed as a way to perform implicitly some sort of deflation with respect to the invariant subspace  $\mathbf{U}_2$  of  $\mathbf{A}$  linked with the largest eigenvalues in  $\mathbf{A}$ , as proposed and discussed for instance in [17, 18, 25], in order to compute those eigenvectors lying in the orthogonal complement of this invariant subspace  $\mathbf{U}_2$ . Chebyshev filtering polynomials, in that respect, do not present the same nice properties as projectors used commonly in such deflation (which have explicit eigenvalues equal to 0 and 1) but still, they can mimic partly these properties and offer an alternative to achieve the same behavior in the Lanczos iteration as with deflation techniques, but without having to compute effectively the basis vectors associated with this deflation.

This combination of Lanczos/Block-Lanczos technique with Chebyshev filtering enables to compute an approximation of all eigenvalues and eigenvectors in a given range  $[\lambda_{min}(\mathbf{A}), \mu[$  without any *a priori* knowledge of their count. Of course, to be efficient, the method relies on the fact that the spectrum of  $\mathbf{A}$  is mostly concentrated in the range  $[\mu, \lambda_{max}(\mathbf{A})]$ , with reasonable value of  $\mu$  so that the number of Chebyshev iterations required to reach a given filtering level  $\varepsilon$  be not too large. In that respect, a preliminary preconditioner applied to  $\mathbf{A}$ , with classical preconditioning techniques, can be also useful to transform and clusterise better the spectrum in the iteration matrix.

Other iterative techniques, like those available in ARPACK [16] for instance, can also be used to approximate a given number of eigenvalues or eigenvectors in the SPD matrix  $\mathbf{A}$ . The difference here is that CHEBYSHEV-PSF targets directly a reduced condition number  $\lambda_{max}/\mu$ , with the risk that, if  $\mu$  is not well located with respect to the actual clustering of the eigenvalues in  $\mathbf{A}$ , the size of the constructed Krylov space can be large. In that respect, the ARPACK equivalent routines target in fact some fixed number of approximate eigenvalues. However, if this predetermined number of

eigenvalues (denoted by  $k$ ) is not large enough with respect to the eigenvalue distribution in  $\mathbf{A}$ , one may be left with a reduced condition number  $\lambda_{max}(\mathbf{A})/\lambda_{k+1}(\mathbf{A})$  that is still rather large. The claim here is not that one method is better than the other, but just that the target is different since with CHEBYSHEV-PSF we control directly the range of eigenvalues, and with the latter technique, one can control better in general the total amount of computations.

### 3 Alternative techniques for the solution of linear systems that exploit the partial spectral decomposition

In this section, we consider solution techniques that exploit the spectral information of the matrix  $\mathbf{A}$ . The common general idea beneath all such techniques, as we will see in detail in the following paragraphs, is to use some precomputed “*near*”-invariant subspace  $\mathbf{W}$  to build a projection onto the orthogonal complement of  $\mathbf{W}$  in some appropriate norm and to exploit such a projector to maintain, implicitly or explicitly all through the iterations, the computed residuals in this orthogonal complement.

In this way, the expectation is that the conjugate gradient method will behave as if the spectrum of  $\mathbf{A}$  was reduced to only those eigenvalues with eigenvectors not lying in the subspace  $\mathbf{W}$ . This is of interest, in particular, if this subspace  $\mathbf{W}$  approximates those eigenvectors associated to the smallest eigenvalues in  $\mathbf{A}$  (as done for instance in § 2) since the reduced condition number of  $\mathbf{A}$  restricted to the orthogonal complement of  $\mathbf{W}$  should be much smaller. In this section, we discuss 5 different solution techniques that exploit this general idea, but in a different manner each.

A general idea can be to decompose the solution of (1) in two parts. The first can be obtained directly with an oblique projection of the right hand side  $\mathbf{b}$  onto the orthogonal complement of  $\mathbf{W}$ . The second part of this solution can then be obtained whether by means of Chebyshev iterations, as in § 3.1, or with the use of conjugate gradient with the first part of this solution as a starting guess, as detailed in § 3.2. These first two techniques make use of the projection with respect to  $\mathbf{W}$  only at the starting point.

The three following techniques make use of such projector explicitly all through the iterations. The first one, as discussed in § 3.3, is to augment the Krylov space generated in the conjugate gradient iterations with the basis  $\mathbf{W}$ , and resumes in a deflation at each iteration of the generated Krylov vectors. The second one, introduced in § 3.4, is to run the conjugate gradient algorithm onto the projected system  $\mathcal{P}\mathbf{A}\mathbf{x} = \mathcal{P}\mathbf{b}$ , where  $\mathcal{P}$  is a projector onto the orthogonal complement of  $\mathbf{W}$  that commutes in some way with  $\mathbf{A}$ . This leads also to compute the solution in two parts, but the use of  $\mathcal{P}\mathbf{A}$  as the iteration matrix maintains explicitly at each iteration the orthogonality of the residuals with respect to  $\mathbf{W}$ . The last technique, detailed in § 3.5, aims at building a preconditionner in  $\mathbf{A}$  that shifts, by adding the value 1, those eigenvalues linked to the “*near*”-invariant subspace  $\mathbf{W}$ , and to run a preconditioned conjugate gradient in the usual way to compute the solution of (1).

Finally, in § 3.6, we introduce algebraic two-grid methods which provide an alternative interpretation of some of the techniques discussed above, and offer the possibility to

derive a family of iterative techniques based on the precomputation of the approximate invariant subspace  $\mathbf{W}$ .

### 3.1 Combination of oblique projection and Chebyshev iteration

In this paragraph, we describe how we use the “*near*”-invariant subspace  $\mathbf{W}$  linked to the smallest eigenvalues to compute solution of linear system with  $\mathbf{A}$ .

The idea is to perform an oblique projection of the initial residual ( $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$ ) onto this “*near*”-invariant subspace in order to get the eigencomponents in the solution corresponding to the smallest eigenvalues, viz.

$$\begin{aligned} \mathbf{r}^{(\text{proj})} &= \mathbf{r}^{(0)} - \mathbf{A}\mathbf{W}\left(\mathbf{W}^T\mathbf{A}\mathbf{W}\right)^{-1}\mathbf{W}^T\mathbf{r}^{(0)}, \\ \text{with } \mathbf{x}^{(\text{proj})} &= \mathbf{x}^{(0)} + \mathbf{W}\left(\mathbf{W}^T\mathbf{A}\mathbf{W}\right)^{-1}\mathbf{W}^T\mathbf{r}^{(0)}. \end{aligned} \quad (8)$$

To compute the remaining part of the solution vector  $\mathbf{A}\tilde{\mathbf{x}} = \mathbf{r}^{(\text{proj})}$ , one can then use the classical Chebyshev algorithm with eigenvalue bounds given by  $\mu$  and  $\lambda_{\max}(\mathbf{A})$ , as explained in § 2.1.

We get the following algorithm, which we call INIT-CHEBYSHEV.

Note that, if  $\mathbf{W}$  exactly spans the invariant subspace generated by  $\mathbf{U}_1$ , the Chebyshev iteration and the oblique projection steps in this solution phase can be performed, though sequentially, in any order *a priori*. However, since  $\text{Span}(\mathbf{W})$  is only an approximation of this invariant subspace, we prefer to perform the Chebyshev step first, followed then by the oblique projection, because this helps to increase the accuracy of the oblique projection by “*minimizing*” the influence of the eigencomponents in  $\mathbf{U}_2$  relative to those in  $\mathbf{U}_1$  in the scalar products  $\mathbf{W}^T\mathbf{r}$  in (8). This is of particular interest when the filtering level  $\varepsilon$  used to compute the basis  $\mathbf{W}$ , as in § 2.2, is not close to machine precision.

The first part in INIT-CHEBYSHEV, from step 1 to 6 corresponds to the application of Chebyshev polynomial in  $\mathbf{A}$  that attempts to reduce by a factor  $\delta$  the eigencomponents in  $\mathbf{r}^{(0)}$  associated with the eigenvalues in the range  $[\mu, \lambda_{\max}(\mathbf{A})]$ , providing thus the resulting residual  $\mathbf{r}^{(m+1)} = \mathcal{F}_{m+1}(\mathbf{A})\mathbf{r}^{(0)}$  and the corresponding iterate approximation  $\mathbf{x}^{(m+1)}$  such that  $\mathbf{b} - \mathbf{A}\mathbf{x}^{(m+1)} = \mathbf{r}^{(m+1)}$ . The factor  $\delta$  is *a priori* the same as the level of filtering  $\varepsilon$  described in § 2.1, except that it may be fixed on purpose to a value different than  $\varepsilon$  in this solution phase.

Steps ii and iii are connected together with the relation that,

$$\mathbf{r}^{(m+1)} = \mathcal{F}_{m+1}(\mathbf{A})\mathbf{r}^{(0)} = \frac{T_{m+1}(\omega_\mu(\mathbf{A}))\mathbf{r}^{(0)}}{T_{m+1}(d_\mu)} = \frac{1}{\sigma_{m+1}}T_{m+1}\left(d_\mu\mathbf{I} - \alpha_\mu\mathbf{A}\right)\mathbf{r}^{(0)}$$

where we note  $d_\mu = \frac{\lambda_{\max} + \mu}{\lambda_{\max} - \mu}$ ,  $\alpha_\mu = \frac{2}{\lambda_{\max} - \mu}$  and  $\sigma_m = T_m(d_\mu)$  for all  $m \geq 0$ . In that respect, step iii can also be replaced by the equivalent following three-term recurrent relation,

$$\mathbf{r}^{(m+1)} = 2\frac{\sigma_m}{\sigma_{m+1}}\left(d_\mu\mathbf{r}^{(m)} - \alpha_\mu\mathbf{A}\mathbf{r}^{(m)}\right) - \frac{\sigma_{m-1}}{\sigma_{m+1}}\mathbf{r}^{(m-1)} \quad (9)$$

ALGORITHM 2: INIT-Chebyshev

**Input:** A “near”-invariant orthonormal basis  $\mathbf{W}$  linked to the all eigenvalues in  $\mathbf{A}$  in the range  $]0, \mu[$

**Begin**

**Chebyshev iteration on the interval  $[\mu, \lambda_{max}(\mathbf{A})]$**

1.  $\alpha_\mu = \frac{2}{\lambda_{max}(\mathbf{A}) - \mu}$  and  $d_\mu = \frac{\lambda_{max}(\mathbf{A}) + \mu}{\lambda_{max}(\mathbf{A}) - \mu}$
2.  $\sigma_0 = 1$  and  $\sigma_1 = d_\mu$
3. Choose an initial guess  $\mathbf{x}^{(0)}$  and  $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$
4.  $\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \frac{\alpha_\mu}{d_\mu} \mathbf{r}^{(0)}$  and  $\mathbf{r}^{(1)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(1)}$
5. **For**  $m = 1, 2, \dots$ , until  $\sigma_m < \delta$  **Do:**
  - i.  $\sigma_{m+1} = 2d_\mu \sigma_m - \sigma_{m-1}$
  - ii.  $\mathbf{x}^{(m+1)} = 2 \frac{\sigma_m}{\sigma_{m+1}} (d_\mu \mathbf{x}^{(m)} + \alpha_\mu \mathbf{r}^{(m)}) - \frac{\sigma_{m-1}}{\sigma_{m+1}} \mathbf{x}^{(m-1)}$
  - iii.  $\mathbf{r}^{(m+1)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(m+1)}$
6. **EndDo**

**Oblique projection onto  $\text{Span}(\mathbf{W})$**

7.  $\mathbf{e}^{(\text{proj})} = \mathbf{W} (\mathbf{W}^T \mathbf{A} \mathbf{W})^{-1} \mathbf{W}^T \mathbf{r}^{(m+1)}$
8.  $\mathbf{x}^{(\text{sol})} = \mathbf{x}^{(m+1)} + \mathbf{e}^{(\text{proj})}$

**End**

that corresponds to the Chebyshev iteration giving  $\mathbf{r}^{(m+1)}$ .

Step 7 consists of solving the error equation associated with  $\mathbf{x}^{(m)}$  in the space spanned by  $\mathbf{W}$ , while step 8 implements the final update/correction.

### 3.2 Conjugate Gradient with an oblique projection as a starting point

As proposed in [1, 21, 23, 28], once an approximation of the invariant subspace associated with the smallest eigenvalues is obtained, we can use it for the computation of further solutions. The idea is to perform an oblique projection of the initial residual onto this invariant subspace in order to get the eigencomponents in the solution corresponding to the smallest eigenvalues, as in (8), and then to perform a classical conjugate gradient to compute the remaining part of the solution vector. Note that we do not project the given matrix in any way, but that we just run the classical conjugate gradient with the original matrix and an initial guess which, we expect, will remove all the troubles that the classical conjugate gradient algorithm would encounter otherwise.

Indeed, in the presence of clusters of eigenvalues at the extreme of the spectrum

of  $\mathbf{A}$ , combined with some ill-conditioning, classical conjugate gradient may show a convergence history of the scaled residuals with sequences of plateaux and sharp drops. These plateaux (see [27]) are well understood and correspond to the discovery of separate eigenvalues within a cluster. The problem is that these plateaux can be rather large, in terms of number of iterations, even when these extreme clusters incorporate only a few eigenvalues. Block conjugate gradient can help to reduce the size of these plateaux, but in general cannot remove them completely (see [1]).

We get the following algorithm, which we call INIT-CG.

ALGORITHM 3: INIT-CG
<p><b>Input:</b> A “near”-invariant orthonormal basis <math>\mathbf{W}</math> linked to the all eigenvalues in <math>\mathbf{A}</math> in the range <math>]0, \mu[</math>.</p> <p><b>Begin</b></p> <ol style="list-style-type: none"> <li>1. <math>\mathbf{x}^{(0)} = \mathbf{W}(\mathbf{W}^T \mathbf{A} \mathbf{W})^{-1} \mathbf{W}^T \mathbf{b}</math></li> <li>2. <math>\mathbf{x}^{(\text{sol})} = \text{Classical CG}(\mathbf{A}, \mathbf{b}, \mathbf{x}^{(0)}, \text{tol})</math>.</li> </ol> <p><b>End</b></p>

### 3.3 The Deflated-Conjugate Gradient

In this subsection, we consider the deflated version of conjugate gradient described in [25]. This algorithm exploits the link between the Lanczos algorithm and the standard conjugate gradient algorithm, and is mathematically equivalent to Nicolaide’s algorithm which is derived from a “deflated” Lanczos procedure [18].

Let  $p$  real vectors  $\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \dots, \mathbf{w}^{(p)}$  be given, along with a unit vector  $\mathbf{v}^{(1)}$  that is orthogonal to  $\mathbf{w}^{(i)}$  for  $i = 1, 2, \dots, p$ . Define  $\mathbf{W} = [\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \dots, \mathbf{w}^{(p)}]$ . We assume that  $[\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \dots, \mathbf{w}^{(p)}]$  is a set of linearly independent vectors. The deflated-CG algorithm builds a sequence  $\{\mathbf{v}^{(k)}\}_{k=1,2,\dots}$  of vectors such that

$$\mathbf{v}^{(k+1)} \perp \text{Span}\{\mathbf{W}, \mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \dots, \mathbf{v}^{(k)}\} \text{ and } \|\mathbf{v}^{(k+1)}\|_2 = 1.$$

Assume an initial guess  $\mathbf{x}^{(0)}$  is given such that  $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)} \perp \mathbf{W}$ . We set  $\mathbf{v}^{(1)} = \mathbf{r}^{(0)} / \|\mathbf{r}^{(0)}\|_2$ . Define

$$\mathcal{K}_{p,k}(\mathbf{A}, \mathbf{W}, \mathbf{r}^{(0)}) = \text{Span}\{\mathbf{W}, \mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \dots, \mathbf{v}^{(k)}\}$$

where the  $\mathbf{v}^{(k)}$  are the generated Krylov vectors at step  $k$ .

At the  $k$ th step of our projection method, we seek an approximate solution  $\mathbf{x}^{(k)}$  with

$$\mathbf{x}^{(k)} \in \mathbf{x}^{(0)} + \mathcal{K}_{p,k}(\mathbf{A}, \mathbf{W}, \mathbf{r}^{(0)}) \tag{10}$$

such that

$$\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)} \perp \mathcal{K}_{p,k}(\mathbf{A}, \mathbf{W}, \mathbf{r}^{(0)}). \quad (11)$$

We get the following algorithm, which we call DEF-CG.

ALGORITHM 4: DEF-CG
<p><b>Input:</b> A “near”-invariant orthonormal basis <math>\mathbf{W}</math> linked to the all eigenvalues in <math>\mathbf{A}</math> in the range <math>]0, \mu[</math>.</p> <p><b>Begin</b></p> <ol style="list-style-type: none"> <li>1. Choose an initial guess <math>\mathbf{x}^{(0)}</math> such that <math>\mathbf{W}^T \mathbf{r}^{(0)} = 0</math>, where <math>\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}</math>.</li> <li>2. Solve <math>\mathbf{W}^T \mathbf{A}\mathbf{W}\gamma^{(0)} = \mathbf{W}^T \mathbf{A}\mathbf{r}^{(0)}</math>, for <math>\gamma</math>.</li> <li>3. Set <math>\mathbf{p}^{(0)} = \mathbf{r}^{(0)} - \mathbf{W}\gamma^{(0)}</math>.</li> <li>4. <b>For</b> <math>k = 1, 2, \dots, m</math>, <b>Do:</b> <ol style="list-style-type: none"> <li>i. <math>\alpha_{k-1} = \frac{\mathbf{r}^{(k-1)T} \mathbf{r}^{(k-1)}}{\mathbf{p}^{(k-1)T} \mathbf{A}\mathbf{p}^{(k-1)}}</math></li> <li>ii. <math>\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} + \alpha_{k-1} \mathbf{p}^{(k-1)}</math></li> <li>iii. <math>\mathbf{r}^{(k)} = \mathbf{r}^{(k-1)} - \alpha_{k-1} \mathbf{A}\mathbf{p}^{(k-1)}</math></li> <li>iv. <math>\beta_{k-1} = \frac{\mathbf{r}^{(k)T} \mathbf{r}^{(k)}}{\mathbf{r}^{(k-1)T} \mathbf{r}^{(k-1)}}</math></li> <li>v. <math>\mathbf{z}^{(k)} = \mathbf{r}^{(k)} - \mathbf{W}(\mathbf{W}^T \mathbf{A}\mathbf{W})^{-1} \mathbf{W}^T \mathbf{A}\mathbf{r}^{(k)}</math></li> <li>vi. <math>\mathbf{p}^{(k)} = \beta_{k-1} \mathbf{p}^{(k-1)} + \mathbf{z}^{(k)}</math></li> </ol> </li> <li>5. <b>EndDo</b></li> </ol> <p><b>End</b></p>

### 3.4 Deflation as a preconditioner

To describe this approach, we define the projection  $\mathcal{P}$  by

$$\mathcal{P} = \mathbf{I} - \mathbf{A}\mathbf{W}(\mathbf{W}^T \mathbf{A}\mathbf{W})^{-1} \mathbf{W}^T, \quad \mathbf{W} \in \mathbb{R}^{n \times p} \quad (12)$$

where  $\text{Span}(\mathbf{W})$  is the deflation subspace, i.e. the space to be projected out of the residual, and  $\mathbf{I}$  is the identity matrix. We assume  $p \ll n$  and that  $\mathbf{W}$  has full rank. We split the solution vector  $\mathbf{x}$  into two parts

$$\mathbf{x} = (\mathbf{I} - \mathcal{P}^T) \mathbf{x} + \mathcal{P}^T \mathbf{x}.$$

The first part  $(\mathbf{I} - \mathcal{P}^T)\mathbf{x}$  is the component of  $\mathbf{x}$  contained in  $\mathbf{W}$ , whereas the second part  $\mathcal{P}^T\mathbf{x}$  is orthogonal to  $\mathbf{W}$ . The first part is computed from:

$$\begin{aligned} (\mathbf{I} - \mathcal{P}^T)\mathbf{x} &= (\mathbf{A}\mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}\mathbf{W}^T)^T\mathbf{x} \\ &= \mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}\mathbf{W}^T\mathbf{A}\mathbf{x} \\ &= \mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}\mathbf{W}^T\mathbf{b}. \end{aligned}$$

To compute the second part  $\mathcal{P}^T\mathbf{x}$  we use  $\mathbf{A}\mathcal{P}^T\mathbf{x} = \mathcal{P}\mathbf{A}\mathbf{x} = \mathcal{P}\mathbf{b}$  (see PROPOSITION 1), and solve  $\mathbf{x}$  from

$$\mathcal{P}\mathbf{A}\mathbf{x} = \mathcal{P}\mathbf{b} \tag{13}$$

Kaasschieter [14] notes that a positive semi-definite system can be solved by CG as long as the right-hand side is consistent (i.e. as long as  $\mathbf{b} = \mathbf{A}\mathbf{x}$  for some  $\mathbf{x}$ ). This is certainly true for (13), where the same projection is applied to both sides of nonsingular system. Furthermore, he notes that, because the null space never enters the iteration, the corresponding zero-eigenvalues do not influence the convergence. We get the following algorithm, which we call PROJ-CG.

ALGORITHM 5: PROJ-CG
<p><b>Input:</b> A “near”-invariant orthonormal basis <math>\mathbf{W}</math> linked to the all eigenvalues in <math>\mathbf{A}</math> in the range <math>]0, \mu[</math>.</p> <p><b>Begin</b></p> <ol style="list-style-type: none"> <li>1. <math>\mathcal{P} = \mathbf{I} - \mathbf{A}\mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}\mathbf{W}^T</math>.</li> <li>2. <math>\mathbf{x}^{(\text{Part1})} = \mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}\mathbf{W}^T\mathbf{b}</math>.</li> <li>3. <math>\mathbf{x}^{(\text{Part2})} = \text{Classical CG}(\mathcal{P}\mathbf{A}, \mathcal{P}\mathbf{b}, \mathbf{x}^{(0)}, \text{tol})</math>.</li> <li>4. <math>\mathbf{x}^{(\text{sol})} = \mathbf{x}^{(\text{Part1})} + \mathcal{P}^T\mathbf{x}^{(\text{Part2})}</math>.</li> </ol> <p><b>End</b></p>

We recall the main properties of this projection  $\mathcal{P}$ , which uses the basis  $\mathbf{W}$  computed by CHEBYSHEV-PSF for the construction of PROJ-CG.

**Proposition 1** The operator  $\mathcal{P}$  has the following properties:

1.  $\mathcal{P}^2 = \mathcal{P}$ ,
2.  $\mathbf{A}\mathcal{P}^T = (\mathcal{P}\mathbf{A})^T = \mathcal{P}\mathbf{A}$ ,
3.  $\mathcal{P}\mathbf{A}\mathcal{P}^T = \mathcal{P}\mathbf{A}$ ,
4. The matrix  $\mathcal{P}\mathbf{A}$  is symmetric positive semi-definite.

5.  $\mathcal{P}\mathbf{A}\mathbf{W} = 0$ .

**Proof**

1.  $\mathcal{P}^2 = \mathbf{I} - 2\mathbf{A}\mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}\mathbf{W}^T + \mathbf{A}\mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}\mathbf{W}^T\mathbf{A}\mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}\mathbf{W}^T = \mathcal{P}$ .
2.  $(\mathcal{P}\mathbf{A})^T = (\mathbf{A} - \mathbf{A}\mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}\mathbf{W}^T\mathbf{A})^T = \mathbf{A} - \mathbf{A}\mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}\mathbf{W}^T\mathbf{A} = \mathcal{P}\mathbf{A}$  (because  $\mathbf{A}^T = \mathbf{A}$ ). Therefore  $\mathcal{P}\mathbf{A}$  is symmetric and  $\mathcal{P}\mathbf{A} = (\mathcal{P}\mathbf{A})^T = \mathbf{A}^T\mathcal{P}^T = \mathbf{A}\mathcal{P}^T$ .
3.  $\mathcal{P}\mathbf{A} = \mathbf{A}\mathcal{P}^T \implies \mathcal{P}\mathbf{A}\mathcal{P}^T = \mathcal{P}^2\mathbf{A} = \mathcal{P}\mathbf{A}$ .
4. By hypothesis,  $0 \leq \mathbf{z}^T\mathbf{A}\mathbf{z}$  for all  $\mathbf{z}$ . In particular,  $0 \leq (\mathcal{P}^T\mathbf{z})^T\mathbf{A}(\mathcal{P}^T\mathbf{z}) = \mathbf{z}^T\mathcal{P}\mathbf{A}\mathcal{P}^T\mathbf{z}$ , so that  $\mathcal{P}\mathbf{A}\mathcal{P} = \mathcal{P}\mathbf{A}$  is positive semi-definite.
5.  $\mathcal{P}\mathbf{A}\mathbf{W} = \mathbf{A}\mathbf{W} - \mathbf{A}\mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}\mathbf{W}^T\mathbf{A}\mathbf{W} = \mathbf{A}\mathbf{W} - \mathbf{A}\mathbf{W} = 0$ .

□

### 3.5 A spectral low-rank update preconditioner

In this paragraph, we consider the approach proposed in [3] that attempts to remove the effect of the smallest eigenvalues in magnitude in the preconditioned matrix, which potentially can slow down the convergence of Krylov solvers. Roughly speaking, the proposed technique consists in solving the preconditioned system exactly on a coarse space and using this information to update the preconditioned residual. We recall now the main components for the construction of this preconditioning technique. Let  $\mathbf{W}$  be set of eigenvectors associated with the set of eigenvalues  $\lambda_i$  such that  $\lambda_i \leq \mu$ .

**Proposition 2** If  $\mathbf{A}$  is SPD, then  $\mathbf{A}_c = \mathbf{W}^T\mathbf{A}\mathbf{W}$  is SPD. The matrix defined by  $\mathbf{M} = \mathbf{I} + \mathbf{M}_c$ , with  $\mathbf{M}_c = \mathbf{W}\mathbf{A}_c^{-1}\mathbf{W}^T$ , is SPD and  $\mathbf{M}\mathbf{A}$  is similar to a matrix whose eigenvalues are

$$\begin{cases} \eta_i = \lambda_i & \text{if } \lambda_i > \mu, \\ \eta_i = \lambda_i + 1 & \text{if } \lambda_i \leq \mu. \end{cases}$$

**Proof**

By construction  $\mathbf{A}_c$  is symmetric, let us show that it is positive definite.  $\mathbf{W}$  is  $n \times k$  matrix. Let  $\mathbf{z} \in \mathbb{R}^k$ ,  $\mathbf{z} \neq 0$ .  $\mathbf{z}^T\mathbf{A}_c\mathbf{z} = \mathbf{z}^T\mathbf{W}^T\mathbf{A}\mathbf{W}\mathbf{z} = (\mathbf{W}\mathbf{z})^T\mathbf{A}\mathbf{W}\mathbf{z} = \|\mathbf{W}\mathbf{z}\|_{\mathbf{A}}$ .  $\mathbf{W}\mathbf{z} \neq 0$  because  $\mathbf{W}$  has full rank. Then  $\mathbf{z}^T\mathbf{A}_c\mathbf{z}$  is greater than 0 because  $\mathbf{A}$  is SPD. Therefore  $\mathbf{A}_c$  is SPD matrix and consequently has full rank.

Let  $\mathbf{x} \in \mathbb{R}^n$ ,  $\mathbf{x} \neq 0$ .  $\mathbf{x}^T\mathbf{M}_c\mathbf{x} = (\mathbf{W}^T\mathbf{x})^T\mathbf{A}_c^{-1}(\mathbf{W}^T\mathbf{x}) = \|(\mathbf{W}^T\mathbf{x})\|_{\mathbf{A}_c^{-1}} \geq 0$  as  $\mathbf{A}_c$  is SPD. Therefore  $\mathbf{M}_c$  is a positive semi-definite matrix and  $\mathbf{M} = \mathbf{I} + \mathbf{M}_c$  is a SPD matrix.

Let  $\mathbf{V} = (\mathbf{W}, \mathbf{Z})$ , where  $\mathbf{Z}$  is the set of  $(n - k)$  eigenvectors associated with eigenvalues  $\lambda_i > \mu$ . Let  $\mathbf{\Lambda}_1 = \text{diag}(\lambda_i)$  with  $\lambda_i \leq \mu$  and  $\mathbf{\Lambda}_2 = \text{diag}(\lambda_i)$  with  $\lambda_i > \mu$ . The following relations hold

$$\begin{aligned} \mathbf{M}\mathbf{A}\mathbf{W} &= \mathbf{A}\mathbf{W} + \mathbf{M}_c\mathbf{A}\mathbf{W} \\ &= \mathbf{W}\mathbf{\Lambda}_1 + \mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}(\mathbf{W}^T\mathbf{A}\mathbf{W}) \\ &= \mathbf{V}_1(\mathbf{\Lambda}_1 + \mathbf{I}_k) \end{aligned}$$

where  $\mathbf{I}_k$  denotes the  $(k \times k)$  identity matrix, and

$$\begin{aligned}\mathbf{MAZ} &= \mathbf{V}_2\boldsymbol{\Lambda}_2 + \mathbf{W}\left(\mathbf{W}^T\mathbf{A}\mathbf{W}\right)^{-1}\mathbf{W}^T\mathbf{A}\mathbf{Z} \\ &= \mathbf{Z}\boldsymbol{\Lambda}_2 + \mathbf{W}\left(\mathbf{W}^T\mathbf{A}\mathbf{W}\right)^{-1}\mathbf{W}^T\mathbf{Z}\boldsymbol{\Lambda}_2 \\ &= \mathbf{Z}\boldsymbol{\Lambda}_2 \quad \text{since } \mathbf{W}^T\mathbf{Z} = 0.\end{aligned}$$

We then have

$$\mathbf{MAV} = [\mathbf{W}, \mathbf{Z}] \begin{bmatrix} (\boldsymbol{\Lambda}_1 + \mathbf{I}_k) & 0 \\ 0 & \boldsymbol{\Lambda}_2 \end{bmatrix}.$$

Therefore,  $\mathbf{MA}$  is similar to  $\begin{bmatrix} (\boldsymbol{\Lambda}_1 + \mathbf{I}_k) & 0 \\ 0 & \boldsymbol{\Lambda}_2 \end{bmatrix}$ , with the smallest eigenvalues shifted by the value one. □

We can then use this spectral preconditioner  $\mathbf{M}$  to run the preconditioned CG, yielding thus the following algorithm which we call SLRU (Spectral Low Rank Update).

ALGORITHM 6: SLRU
<p><b>Input:</b> A “near”-invariant orthonormal basis <math>\mathbf{W}</math> linked to the all eigenvalues in <math>\mathbf{A}</math> in the range <math>]0, \mu[</math>.</p> <p><b>Begin</b></p> <ol style="list-style-type: none"> <li>1. <math>\mathbf{M} = \mathbf{I} + \mathbf{W}\left(\mathbf{W}^T\mathbf{A}\mathbf{W}\right)^{-1}\mathbf{W}^T</math>.</li> <li>2. <math>\mathbf{x}^{(\text{sol})} = \text{Preconditioned CG}\left(\mathbf{A}, \mathbf{b}, \mathbf{x}^{(0)}, \mathbf{M}, \text{tol}\right)</math>.</li> </ol> <p><b>End</b></p>

### 3.6 Algebraic two-grid cycle

The multigrid methods which are among the fastest techniques to solve a linear system (1) arising from the discretization of elliptic partial differential equations [11]. The core of the multigrid algorithms is a two-grid procedure that is applied recursively. A classical two-grid cycle can be shortly described as follows. On the fine grid a few iterations of a smoother is applied that attempts to reduce the high frequencies of the error that the components of the error in the space spanned by the vectors associated with the largest eigenvalues of  $\mathbf{A}$ . The residual is then projected on the coarse grid where the low frequencies, that are the components associated with the smallest eigenvalues, can be captured and the coarse error equation is solved. The coarse error is prolonged back to the fine grid to update the approximation computed by the pre-smoother and a few more steps of the smoother is applied. Finally, if the new iterate is not accurate enough a two-grid cycle is applied iteratively. In classical multigrid, the coarse space is not defined explicitly through the knowledge of the eigenspaces but by the selection of a space that is expected to capture them. In this work, we explicitly define the coarse

<b>ALGORITHM 7: GENERIC TWO-GRID CYCLE</b>
--

**Input:** A “*near*”-invariant orthonormal basis  $\mathbf{W}$  linked to the all eigenvalues in  $\mathbf{A}$  in  $\mathbf{A}$  in the range  $]0, \mu[$ .

**Begin**

1. set  $\mathbf{x}^{(1)} = 0$
2. **For**  $k=1, iter$  **Do:**

**Pre-smoothing: damp the high frequencies of the error**

i.  $\mathbf{x}_1^{(k)} = \mathbf{S}(\mathbf{x}^{(k)}, \mathbf{b}, \xi)$ .

**Coarse grid correction**

ii.  $\mathbf{x}_2^{(k)} = \mathbf{x}_1^{(k)} + \mathbf{W}\mathbf{A}_c^{-1}\mathbf{W}^T(\mathbf{b} - \mathbf{A}\mathbf{x}_1^{(k)})$ .

**Post-smoothing: damp again the high frequencies of the error**

iii.  $\mathbf{x}^{(k+1)} = \mathbf{S}(\mathbf{x}_2^{(k)}, \mathbf{b}, \xi)$ .

3. **EndFor**
4.  $\mathbf{x} = \mathbf{x}^{(iter)}$

**End**

space by computing the eigenvectors  $\mathbf{W}$  associated with the smallest eigenvalues of  $\mathbf{A}$ . For symmetric problems, the restriction operator is denoted  $\mathbf{R} = \mathbf{W}^T$  and the prolongation operator is its transpose. The matrix associated with the coarse grid error problem is defined by a Galerkin formula  $\mathbf{A}_c = \mathbf{R}\mathbf{A}\mathbf{R}^T$ .

A generic two-level multigrid cycle is illustrated in ALGORITHM 7, where  $\mathbf{S}(\mathbf{x}^{(0)}, \mathbf{b}, \xi)$  defines the smoother that performs  $\xi$  iterations on the initial guess  $\mathbf{x}^{(0)}$  to solve  $\mathbf{A}\mathbf{x} = \mathbf{b}$ .

## 4 Numerical experiments

In this section, we show some numerical results that illustrate the effectiveness of combining the “partial spectral decomposition” with the different approaches for the solution of sparse symmetric and positive definite (SPD) linear systems, as described in the previous sections.

As we will see in the following, the experiments show that the different algorithms exhibit very similar numerical behaviour. This is particularly true when the precomputed “*near*”-invariant basis  $\mathbf{W}$  is obtained with an accuracy close to machine pre-

cision. In this case, this numerical behaviour corresponds to the one that we would obtain in general with a linear system where the smallest eigenvalue is given by the cut-of value  $\mu$  in the CHEBYSHEV-PSF algorithm, with resulting reduced condition number of  $\lambda_{max}(\mathbf{A})/\mu$ . The purpose of the following sections is to analyse experimentally this in details, and in particular to investigate the impact of much less accurate precomputed  $\mathbf{W}$  basis on the actual behavior of these different algorithms.

For all the numerical experiments reported in this section, a first level of left preconditioner  $\mathbf{M}_1 = \mathbf{L}\mathbf{L}^T$  is constructed using Incomplete Cholesky factorization IC( $t$ ) of  $\mathbf{A}$  with  $t$  as the threshold. The purpose of this first preliminary preconditioning of the given linear system is to better cluster the eigenvalues so that the invariant subspace associated with the smallest eigenvalues be of small dimension. We denote by  $\widehat{\mathbf{b}} = \mathbf{L}^{-1}\mathbf{b}$  and  $\widehat{\mathbf{A}} = \mathbf{L}^{-1}\mathbf{A}\mathbf{L}^{-T}$  the matrix symmetrically preconditioned with IC( $t$ ) which is SPD and similar to the matrix  $\mathbf{M}_1^{-1}\mathbf{A}$ . All experiments were performed in 64 bits floating-point arithmetic using MATLAB.

The stopping criterion used in these tests is the backward error  $\|\widehat{\mathbf{b}} - \widehat{\mathbf{A}}\mathbf{x}^{(i)}\|_2/\|\widehat{\mathbf{b}}\|_2$ . In all our runs, we have monitored this backward error down to machine precision when possible to illustrate and study the complete convergence history of the schemes. The initial guess is always  $\mathbf{x}^{(0)} = 0$  and the right hand side is chosen so that the solution of the initial linear system is the vector of all ones. The block size  $s$  is set to 1 and the cut-off eigenvalue  $\mu$  to  $\lambda_{max}(\widehat{\mathbf{A}})/10$ .

Name	Size	Short description
BCSSTK27	1224	Dynamic analyses in structural engineering - Buckling analysis
BCSSTK14	1806	Static analyses in structural engineering - Roof of the Omni Coliseum, Atlanta
BCSSTK15	3948	from the BCSSTRUC2 group of the Harwell-Boeing Collection
BCSSTK16	4884	Static analyses in structural engineering - U.S. Army Corps of Engineers dam
S1RMQ4M1	5489	Structural mechanics - Cylindrical shell
PDE 1	6084	2D Poisson diffusion equation discretized by uniform mesh in a square region
PDE 2	7969	2D heterogenous diffusion equation discretized by finite element in a L shape region

Table 1: *set of SPD test matrix*

In Table 1 we summarize for each matrix, the size and the application in which this matrix arises. In particular, matrices BCSSTK are extracted from the Harwell-Boeing collection [7] and S1RMQ4M1 from Matrix Market.

#### 4.1 Distribution of eigenvalues

In Table 2 and Figure 2, we illustrate the effect of the Incomplete Cholesky (IC) factorization on the eigenvalue distribution of the preconditioned matrix  $\widehat{\mathbf{A}}$ . As mentioned

before, a clustered spectrum of the preconditioned matrix is usually considered a desirable property for fast convergence of Krylov solvers and makes the numerical techniques considered in this paper affordable because only a few eigencomponents have to be handled by a complementary approach.

Matrix	Unpreconditioned matrix		Preconditioned matrix			Number of eigenvalues below the $\mu = \lambda_{max}/10$
	Smallest eig	Largest eig	Threshold IC(t)	Smallest eig	Largest eig	
BCSSTK27	$1.43 \cdot 10^2$	$3.46 \cdot 10^6$	$10^{-2}$	$4.48 \cdot 10^{-2}$	3.49	8
BCSSTK14	1	$1.19 \cdot 10^{10}$	$10^{-2}$	$1.41 \cdot 10^{-2}$	3.13	23
BCSSTK15	1	$6.53 \cdot 10^9$	$10^{-3}$	$5.35 \cdot 10^{-3}$	1.77	4
BCSSTK16	1	$4.94 \cdot 10^9$	$10^{-2}$	$9.62 \cdot 10^{-2}$	1.33	3
S1RMQ4M1	$3.79 \cdot 10^{-1}$	$6.87 \cdot 10^6$	$10^{-2}$	$7.15 \cdot 10^{-4}$	6.06	3 <sup>1</sup>
PDE 1	$3.16 \cdot 10^{-3}$	7.99	$10^{-2}$	$2.86 \cdot 10^{-2}$	1.16	4
PDE 2	$2.92 \cdot 10^{-9}$	2.17	$10^{-2}$	$1.79 \cdot 10^{-7}$	1.11	3

Table 2: *Distribution of eigenvalues*

**Eigenvalue Distribution of BCSSTK14's Matrix preconditioned with IC( $10^{-2}$ )**

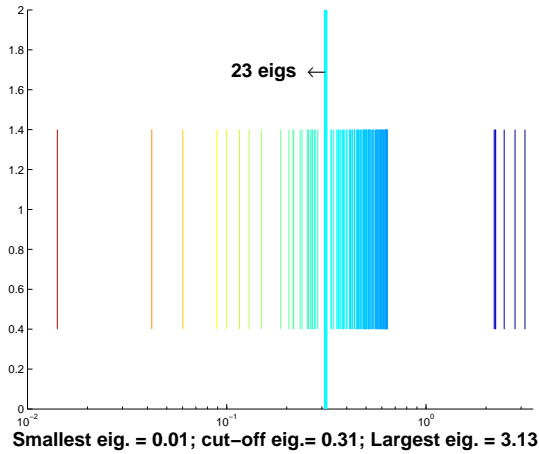


Figure 2: *The eigenvalue distribution of the preconditioned matrix*

Figure 2 shows the spectrum of BCSSTK14's matrix, with the particularity that only the 100 smallest and 6 largest eigenvalues are displayed (the others have not been

<sup>1</sup>In this case we use  $\mu = \lambda_{max}/200$ .

computed). We first notice the high classical condition number of order  $10^{10}$  for the original matrix, since the smallest eigenvalue is 1 and the largest is  $1.19 \cdot 10^{10}$ , and we can also observe that the condition number of the preconditioned matrix is reduced to about  $10^2$ .

## 4.2 The cost of precomputing the orthonormal basis $\mathbf{W}$

The partial spectral factorization (described in § 2) depends on three different parameters which correspond to the choice of the starting block size  $s$ , of the cut-off eigenvalue  $\mu$ , and of the filtering level  $\varepsilon$  under which we try to maintain by means of Chebyshev filtering those eigencomponents relative to all eigenvalues bigger than  $\mu$ .

The value of  $s$  influences mostly the number of Block-Lanczos steps, which will be roughly about  $k/s$ ,  $k$  being equal to the number of eigenvalues outside the damping interval  $[\mu, \lambda_{\max}(\hat{\mathbf{A}})]$  and corresponding to the dimension  $k$  of the Krylov basis generated at the end of the algorithm. In [2], M. Arioli and D. Ruiz did not observe a big impact when varying  $s$  on the actual total number of matrix-vector products induced by the refiltering steps in the Block-Lanczos/Orthodir iterations. Basically, what they observed is that, before reaching convergence, the sum over the iterations of the number of refiltering steps times the block size  $s$  does not vary a lot with different values of  $s$ . It is only at the last step, when convergence is reached, that the block size  $s$  may have some impact, simply because we must filter down to the level  $\varepsilon$  again all the  $s$  vectors in the last block, and the larger is  $s$ , the larger is the amount of work in this last refiltering step. In that respect we set  $s = 1$  for those numerical experiments considered in this paper.

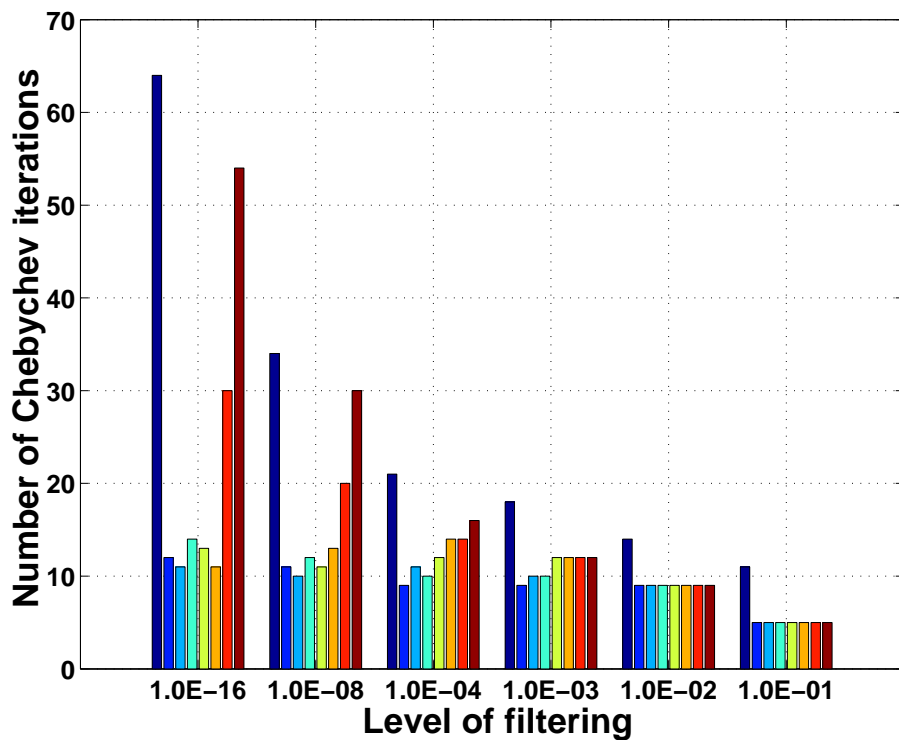
The second parameter,  $\mu$ , splits the spectrum of the matrix  $\hat{\mathbf{A}}$  in two subsets and determines the dimension  $k$  of the invariant subspace that will be computed in the partial spectral factorization phase, depending of course on the actual eigenvalue distribution in the matrix  $\hat{\mathbf{A}}$ . It also determines the convergence rate of the classical Chebyshev iterations that will be performed, both when filtering the Lanczos vectors and when computing the solution. The rapid change in the Chebyshev rate of convergence with smaller values of  $\mu$  induces much more Chebyshev filtering steps at each iteration, and this could only be counterbalanced by a very strong reduction in the total number of iterations in the first phase of the Algorithm. In other words, it is worth reducing the value of  $\mu$  only if there is a very strong clustering of eigenvalues in the spectrum of the iteration matrix  $\hat{\mathbf{A}}$  and if the change in  $\mu$  helps to reduce by a large amount the dimension of the invariant subspace that will be approximated.

The choice of the level of filtering  $\varepsilon$  influences directly the numerical quality of the computed “near”-invariant basis  $\mathbf{W}$  in the CHEBYSHEV-PSF algorithm. To illustrate this, we report in Table 3 the number of Chebyshev filtering iterations performed at each Lanczos step in the CHEBYSHEV-PSF algorithm, and this for different values of the filtering level  $\varepsilon$ . Similarly, in the bar chart in Figure 3, we report the same information for matrices BCSSTK27 and BCSSTK14 which require larger number of Lanczos steps (8 and 23 respectively). Concerning the number of Chebyshev filtering iterations, we can observe in Figure 3 and Table 3 that these mostly differ when varying  $\varepsilon$  in the first and last Lanczos steps, and that during the intermediate stages when building the

Matrix	Cut-off value $\mu$	#eigs $< \mu$	Value of filtering level $\varepsilon$	Number of Chebyshev filtering iterations at steps 4 and v in CHEBYSHEV-PSF			
				Step 4	Step v		
				$k = 0$	$k = 1$	$k = 2$	$k = 3$
BCSSTK15	$(\mu = \frac{\lambda_{max}(\mathbf{A})}{10})$	4	$10^{-16}$	64	13	15	41
			$10^{-8}$	36	11	12	30
			$10^{-4}$	22	10	11	15
			$10^{-3}$	18	10	10	12
			$10^{-2}$	15	9	9	9
			$10^{-1}$	12	5	5	5
BCSSTK16	$(\mu = \frac{\lambda_{max}(\mathbf{A})}{10})$	3	$10^{-16}$	64	20	30	—
			$10^{-8}$	36	14	15	—
			$10^{-4}$	22	6	11	—
			$10^{-3}$	19	4	9	—
			$10^{-2}$	15	3	7	—
			$10^{-1}$	12	2	5	—
SIRMQ4M1	$(\mu = \frac{\lambda_{max}(\mathbf{A})}{200})$	3	$10^{-16}$	296	155	255	—
			$10^{-8}$	166	104	135	—
			$10^{-4}$	101	70	70	—
			$10^{-3}$	85	54	54	—
			$10^{-2}$	68	38	38	—
			$10^{-1}$	52	22	22	—
PDE 1	$(\mu = \frac{\lambda_{max}(\mathbf{A})}{10})$	4	$10^{-16}$	64	16	18	43
			$10^{-8}$	36	10	16	26
			$10^{-4}$	22	8	12	13
			$10^{-3}$	19	7	9	11
			$10^{-2}$	15	6	7	9
			$10^{-1}$	12	5	5	5
PDE 2	$(\mu = \frac{\lambda_{max}(\mathbf{A})}{10})$	3	$10^{-16}$	65	19	24	—
			$10^{-8}$	37	18	15	—
			$10^{-4}$	22	15	11	—
			$10^{-3}$	19	12	11	—
			$10^{-2}$	15	9	9	—
			$10^{-1}$	12	5	5	—

Table 3: Comparison of the number of Chebyshev filtering steps for different values of the filtering level. (Block Lanczos with block size 1).

(a) *BCSSTK27*



(b) *BCSSTK14*

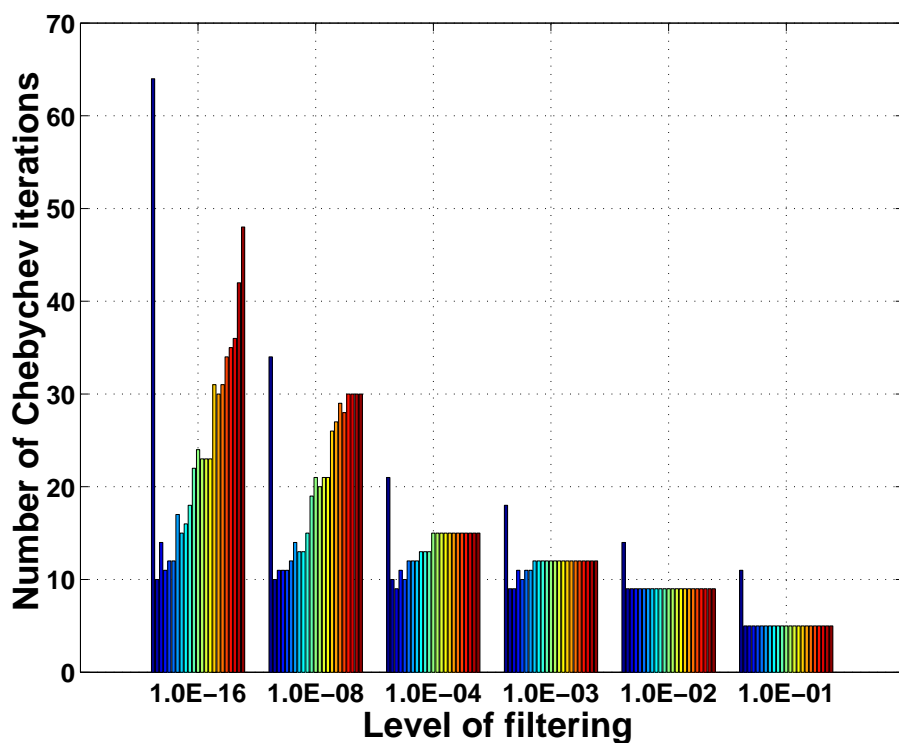


Figure 3: *Number of Chebyshev iterations at each Lanczos step during the partial spectral factorization*

“near”-invariant subspace, these do not vary much with the choice of  $\varepsilon$ . Indeed, the intermediate Chebyshev iterations simply aim at recovering some potential increase in the level of filtering when orthogonalizing the current Lanczos vectors with respect to the previous ones. This is inherent to the Lanczos iteration and does not depend much on the level of the filtering applied to the initial Lanczos basis.

The number of iterations in the first filtering step is obviously directly linked to the choice of the filtering level  $\varepsilon$  since its purpose is to filter some random starting set of vectors under that level. At the final stages also, when convergence or “near”-invariance with the respect to subspace associated with the smallest eigenvalues of  $\hat{\mathbf{A}}$  is reached in the partial spectral factorization phase, some vectors in the last computed set may become strongly colinear to the orthogonal complement of this invariant subspace, implying thus more Chebyshev filtering iterations. These are the two reasons why small values for  $\varepsilon$  imply more Chebyshev iterations in the first and last filtering stages.

### 4.3 Comparison of the different techniques for the solution phase

In this subsection, we compare the numerical convergence behavior of the various algorithms described in § 3. We discuss their numerical efficiency, computational complexity and sensitivity to the accuracy of the spectral information. We mention beforehand that, when the basis  $\mathbf{W}$  is computed with a filtering level close to machine precision ( $10^{-16}$ ), all algorithms presented in § 3 exhibit the same linear rate of convergence. This linear rate of convergence obtained, which can be observed on the convergence histograms in Figures 4 and 5, is directly connected with the cut-off value  $\mu$  that determines a priori the reduced condition number of the linear system that is solved. This is in agreement with the theory on the numerical behavior of the conjugate gradient algorithm.

In Figure 4, we compare the convergence history of the standard conjugate gradient versus INIT-CG and SLRU obtained with a basis  $\mathbf{W}$  computed with a level of filtering  $\varepsilon$  equal to  $10^{-16}$ . The backward error of the standard conjugate gradient method tends to decrease very rapidly in the first few iterations, then decreases more slowly in an intermediate phase during which oscillations may also appear, and finally stagnates. As opposed to this, INIT-CG and SLRU present a convergence history that coincide almost completely, except perhaps after stagnation.

DEF-CG and PROJ-CG present also the same type of convergence, at least in the first part of their convergence histograms as illustrated by the results in Figure 5. However, we can observe that these are also subject to some numerical instability, in particular after reaching the stagnation phase. Theoretically, the residuals  $\mathbf{r}^{(k)}$  are supposed to stay orthogonal to all the columns of  $\mathbf{W}$ , but in practice, this orthogonality is gradually lost as the iterations progress. Figure 5 shows, for matrix BCSSTK14, for instance, the histogram of

$$\text{ortho}(k) = \max_{j=1,2,\dots,p} \left( \frac{\mathbf{w}^{(j)\top} \mathbf{r}^{(k)}}{\|\mathbf{w}^{(j)}\|_2 \|\mathbf{r}^{(k)}\|_2} \right),$$

computed at each iteration  $k$  in DEF-CG and PROJ-CG. Both curves of the function  $\text{ortho}(k)$  show that loss of orthogonality is inversely proportional to the convergence

of the backward error. One remedy to recover orthogonality as mentioned in [25] is to add the reorthogonalization step

$$\mathbf{r}^{(k)} = \mathbf{r}^{(k)} - \mathbf{W} \left( \mathbf{W}^T \mathbf{W} \right)^{-1} \mathbf{W}^T \mathbf{r}^{(k)} \quad (14)$$

right after  $\mathbf{r}^{(k)}$  is computed in DEF-CG and PROJ-CG. The computational cost of this extra step is of order  $\mathcal{O}(k * n)$ .

The convergence curves with the additional correction of  $\mathbf{r}^{(k)}$  are plotted with a (○, solid line) in Figure 5. In this case, the convergence history of DEF-CG and PROJ-CG are very much close to that of of INIT-CG and SLRU as in Figure 4, with no more

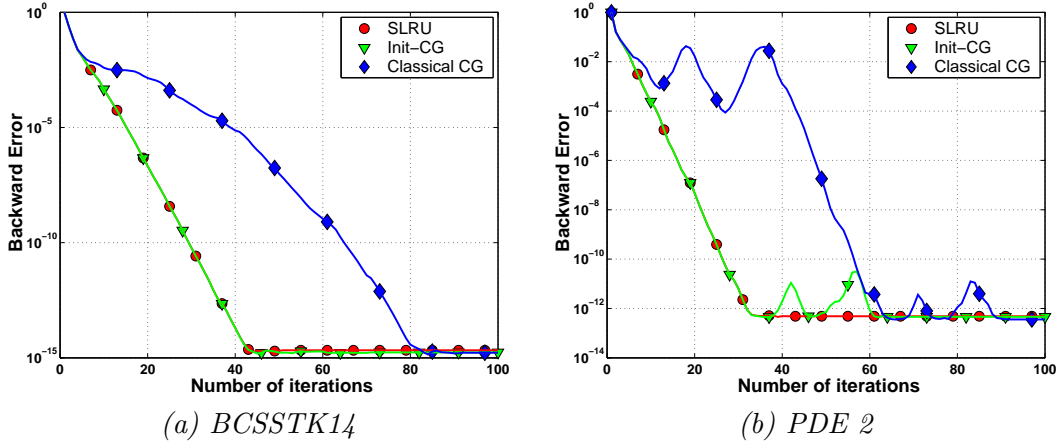


Figure 4: Convergence history for a level of filtering  $\varepsilon$  equal to  $10^{-16}$

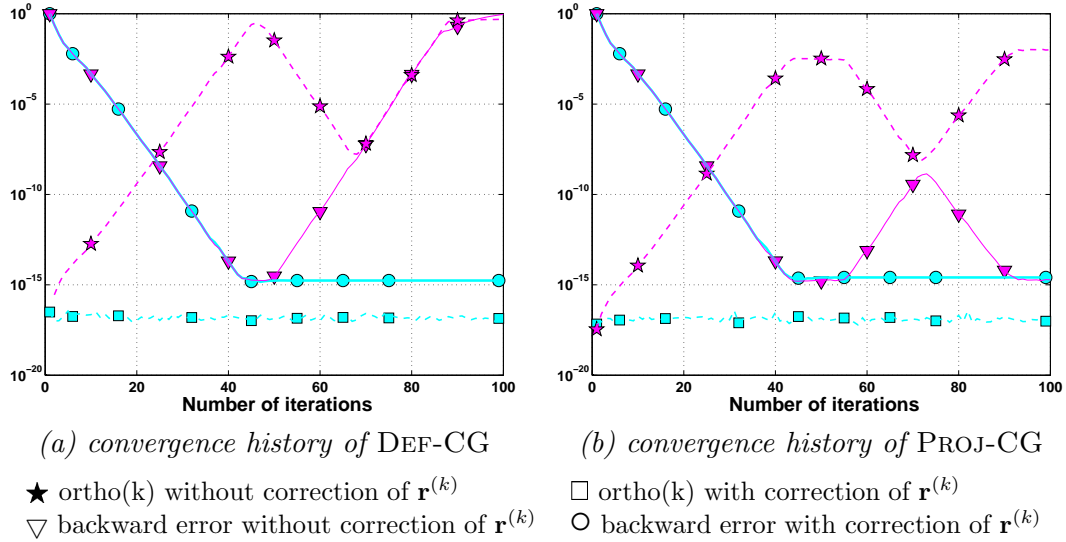


Figure 5: Orthogonality of  $\mathbf{r}^{(k)}$  with respect to  $\mathbf{W}$  precomputed with a level of filtering  $\varepsilon$  equal to  $10^{-16}$  and convergence histories with or without the correction of  $\mathbf{r}^{(k)}$  on the BCSSTK14 matrix.

instability after convergence is reached.

#### 4.4 Practical considerations

We first consider operations count. The initial guess is the most time-consuming overhead. Both computations of  $\mathbf{x}^{(0)}$  and  $\mathbf{p}^{(0)}$  are BLAS2 type operations (projection onto a subspace of size  $p$ ). On the other hand, each iteration, adds merely one dot-product and one vector update.

Let us examine now the memory requirements of these four solution techniques (INIT-CG, DEF-CG, PROJ-CG, and SLRU). All these algorithms require about the same amount of extra storage, of order  $\mathcal{O}(n * p)$ , to store  $\mathbf{W}$  the basis of the “near”-invariant subspace. They also all make use of the same type of operator

$$\mathbf{W}(\mathbf{W}^T \mathbf{A} \mathbf{W})^{-1} \mathbf{W}^T.$$

The only difference is that DEF-CG, PROJ-CG, and SLRU, use this operator at each iteration whereas INIT-CG does exploit this only at the beginning (for computing a starting vector  $\mathbf{x}^{(0)}$ ). The two-grid algorithm is a trade-off as the use the coarse grid correction every other  $\xi$  steps of the smoothers. With respect to this numerical kernel, INIT-CG is the algorithm of choice because it is less expensive, since it avoids the use of oblique projectors within the conjugate gradient iterations explicitly. But the good behavior of INIT-CG illustrated above is due to the fact that the basis  $\mathbf{W}$  used for deflation is computed with an accuracy close to machine precision. As opposed to these last three methods, and this will be illustrated in the following, INIT-CG may however suffer from mis-convergence behavior if the computation of the “near”-invariant basis  $\mathbf{W}$  is much less accurate.

#### 4.5 When the near-invariant subspace basis $\mathbf{W}$ is not very accurate

When the basis vectors  $\mathbf{W}$  are computed with a worse accuracy, as for instance when the level of filtering in CHEBYSHEV-PSF is around  $\varepsilon = 10^{-2}$  or  $10^{-1}$ , the “near”-invariance is lost and the oblique projection in the first step of INIT-CG does not improve the speed of convergence of INIT-CG which eventually behaves like a classical conjugate gradient.

Figure 6 shows the impact of varying the filtering level  $\varepsilon$  (when computing the “near”-invariant  $\mathbf{W}$  basis) on the convergence of INIT-CG. For example, when the level of filtering  $\varepsilon$  is equal to  $10^{-8}$ , the phenomenon of plateaux occurs again, but at intermediate levels. Indeed, INIT-CG (Classical CG with initial oblique projection) shows two different phases in its numerical behavior. The first stage follows very closely the convergence history observed in Figure 4 when the computation of the  $\mathbf{W}$  basis is very accurate. Then, the speed of convergence is disrupted at an intermediate level of the backward error closely related to the value of the filtering level  $\varepsilon$ , and the convergence history resembles again that of the classical conjugate gradient on the original system. The reasons for this is that INIT-CG provides an initial residual to the conjugate gradient that has larger eigencomponents in the orthogonal complement

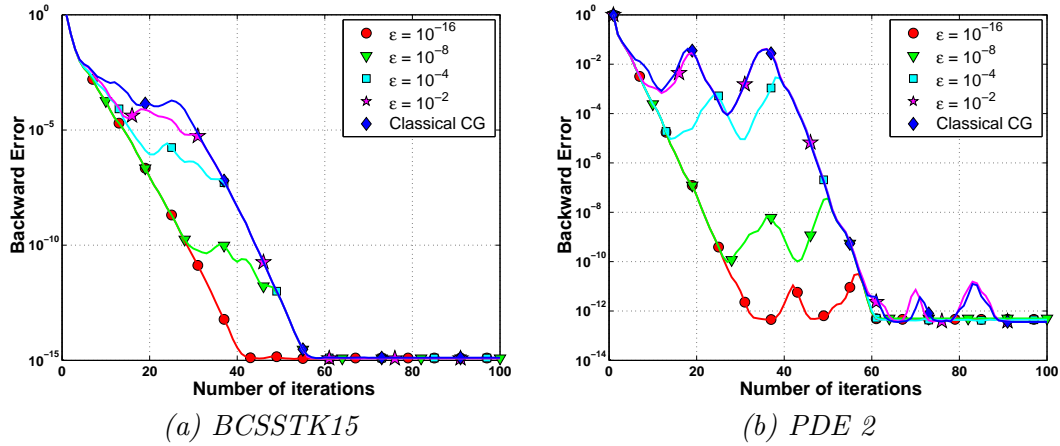


Figure 6: *Convergence history of INIT-CG for different values of level of filtering  $\varepsilon$*

of the basis  $\mathbf{W}$  and eigencomponents of order  $\varepsilon$  in the  $\mathbf{W}$  subspace. Then, after the residual norm has been reduced to about the value of  $\varepsilon$ , the conjugate gradient is left again with a Krylov direction with eigencomponents all of about the same magnitude, and then exhibits the same type of convergence behavior linked to the ill-conditioning of the given matrix. For efficiency, it is clear that one should then stop the iterations in INIT-CG at a backward error about the filtering level  $\varepsilon$  before reaching this intermediate plateau.

The convergence history of DEF-CG and SLRU depicted in Figure 7 for different values of the level of filtering  $\varepsilon$  shows, and this is also mentioned in [25, 29], that these two algorithms exhibit very close numerical behaviors. We can observe anyway that if the accuracy in the computation of the invariant subspace  $\mathbf{W}$  is strongly deteriorated, as with a value  $\varepsilon = 10^{-2}$  for instance, then the speed of convergence may vary slightly. Still, as opposed to INIT-CG, these algorithms show a constant numerical behavior close to the one observed in Figures 4 and 5, even when the filtering level is in the range  $10^{-8}$  to  $10^{-2}$ , as illustrated in Figure 7. This is the benefit of performing at each

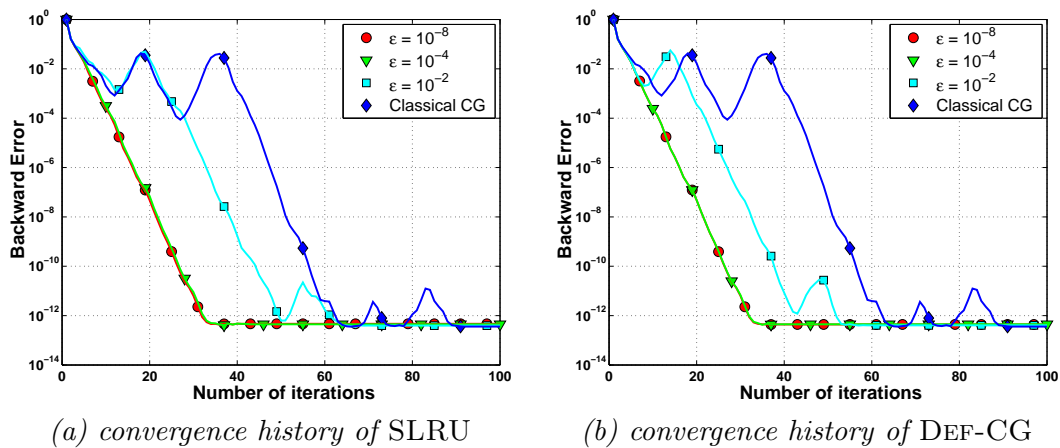


Figure 7: *Convergence history for different values of  $\varepsilon$  on PDE 2*

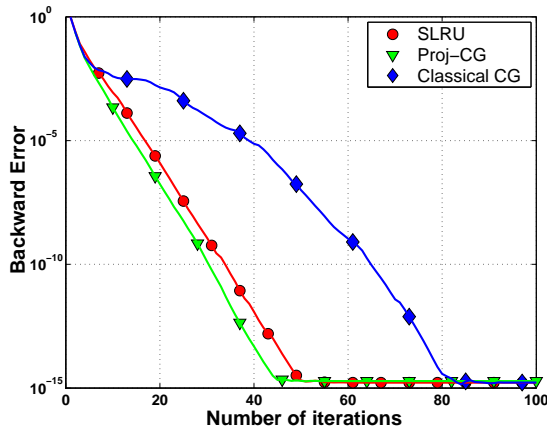


Figure 8: *Convergence history for a level of filtering  $\varepsilon$  equal to  $10^{-2}$  on BCSSTK14*

iteration of the conjugate gradient an oblique projection with the  $\mathbf{W}$  basis that maintains all through the iterations a fixed gap between the eigencomponents corresponding to the smallest eigenvalues and the others.

Figure 8 shows the numerical behavior of PROJ-CG and SLRU when the level of filtering  $\varepsilon$  is large. We can see that the Conjugate Gradient method applied to the projected system ( $\mathcal{P}\mathbf{A}\mathbf{x} = \mathcal{P}\mathbf{b}$ , PROJ-CG) converges a little faster than the conjugate gradient method applied to the preconditioned system via spectral low-rank update (SLRU). As mentioned in [17] a possible explanation of this can be that the effective condition number of the projected system is always below the condition number of the system preconditioned by coarse grid correction, for all matrices  $\mathbf{W} \in \mathbb{R}^{n \times p}$ .

#### 4.6 Combination of oblique projection and Chebyshev iteration

We have postponed in this last section the discussion concerning the results for INIT-CHEBYSHEV since it does not belong to Krylov techniques as the others. INIT-CHEBYSHEV basically corresponds to the juxtaposition of a Chebyshev iterative solver and an oblique projection that act independently on two separate subparts in the spectrum of  $\mathbf{A}$ . In that respect, it can be either viewed as a direct method if the eigenspace is computed very accurately or as a form of two-space cycle, where the smoother is defined by a few steps of Chebyshev iteration.

For sake of comparison with the results shown for the other Krylov based iterative solution techniques, we present in Figure 9 the computed backward error obtained after each Chebyshev iteration combined directly with the application of the final oblique projection. Of course, this is not the way INIT-CHEBYSHEV should work, since the oblique projection should be applied only once at the end, after the termination of the Chebyshev iterations.

In Figure 9, we illustrate the effect of varying the filtering level  $\varepsilon$  (when computing the invariant basis  $\mathbf{W}$ ) on the convergence behavior of INIT-CHEBYSHEV. When the level of filtering  $\varepsilon$  is equal to  $10^{-16}$ , INIT-CHEBYSHEV behaves like the other algorithms,

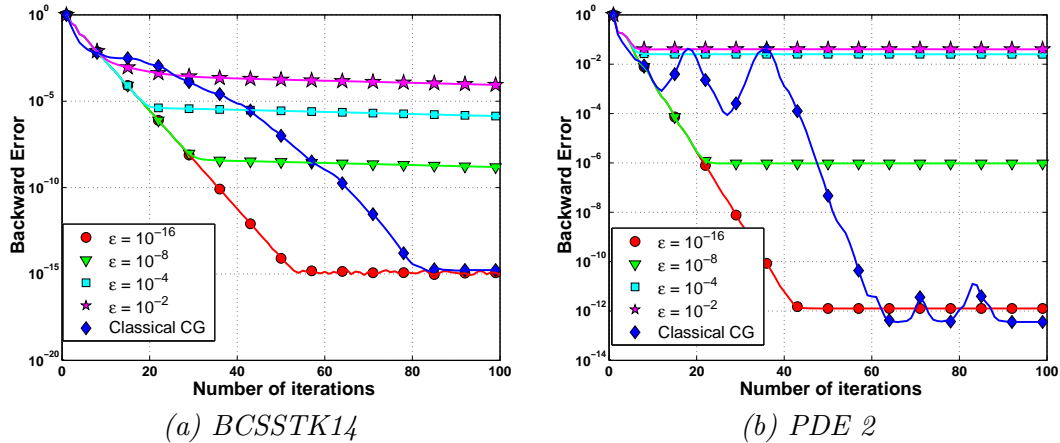


Figure 9: Convergence history of INIT-CHEBYSHEV for different values of level of filtering  $\epsilon$

as illustrated in the previous sections, but with a little difference in the number of iterations. This is simply due to the different rates of convergence between Chebyshev semi-iterative method the Conjugate Gradient method in general.

However, for larger values of the filtering level  $\epsilon$ , we can observe that the convergence history of INIT-CHEBYSHEV is disrupted with a stagnation of the backward error at a level close to the actual value of  $\epsilon$ . The reason for this is simply that the “near”-invariant basis  $\mathbf{W}$  has eigenvectors of order  $\epsilon$  with respect to those eigenvectors corresponding to eigenvalues in the range  $[\mu, \lambda_{max}]$ . Obviously, this level of information is incorporated explicitly in the oblique projection onto this subspace that is applied as a final stage, and perturbs the eigenvectors in the final residual at about that level. For a more detailed analysis of this phenomenon, we refer to [2].

It is also worth mentioning that this combination of oblique projection and Chebyshev iteration is based only on kernels commonly used in iterative methods, that enable us to keep the given ill-conditioned matrix in implicit form. It requires only matrix-vector products plus some vector updates, but no dot-products, as opposed to the other Krylov based solution techniques. This remark is also of some importance in the context of parallel computing, and in particular in distributed memory environments where the computation of the dot product is a well-known bottleneck.

The INIT-CHEBYSHEV can also be viewed as one step of a two-grid cycle where the smoother, that is Chebyshev iterations, is run until convergence. In the next section we will further investigate this point of view when the spectral information is not computed with full accuracy.

#### 4.7 Algebraic two-grid

As mentioned in Section 3.6, one way to exploit the spectral information is to define a two-grid cycle that is run as a stationary iterative solver. This approach only makes sense when the spectral information is not computed very accurately. In that latter case, the schemes based on a deflation to define the initial guess should be preferred

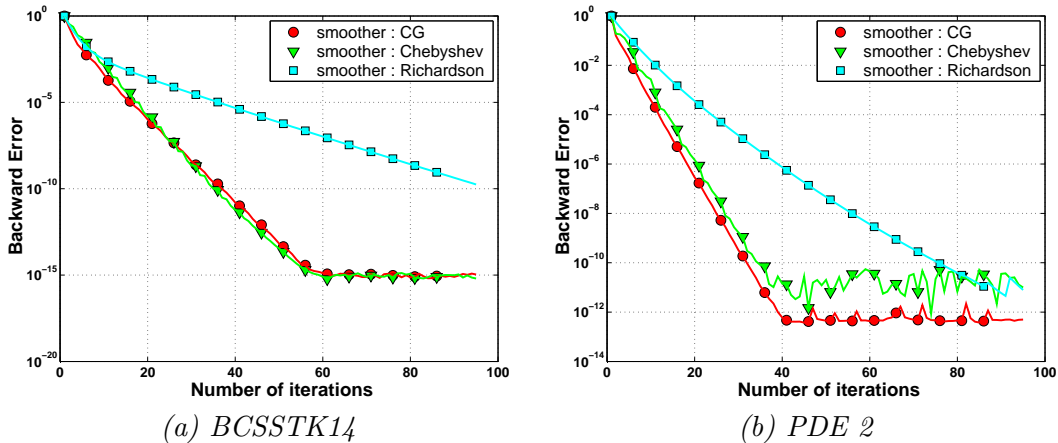


Figure 10: *Convergence history of the two-grid cycle solver with different smoothers using a level of filtering  $\varepsilon = 10^{-4}$*

because they only require one oblique projection. For this reason, we only consider “near”-invariant eigenspaces in this section. To illustrate the numerical behaviour of these schemes we depict in Figure 10 the convergence history of this algebraic two-grid solver using different smoothers; more precisely for these experiments we implement a few steps of CG, of Chebyshev or of damped Richardson. For the Richardson’s iteration we select  $\alpha_{opt} = 2/(\lambda_{max} + \mu)$  that minimizes the spectral radius of the iteration matrix that is equal to  $\max\{|1 - \alpha \lambda_{max}|, |1 - \alpha \mu|\}$ , assuming that the coarse grid correction has removed the effect of eigenvalues in the range  $[\lambda_{min}, \mu]$ . For the experiments illustrated in Figure 10 we consider  $\xi = 5$  smoothing steps. It can be seen that the three solvers exhibit a linear rate of convergence and that CG and Chebyshev have the same smoothing properties.

For sake of efficiency one might want to use a value of  $\xi$  (number of smoothing steps) as large as possible in order to reduce the number of cycles, and consequently the number of oblique projections, while preserving the same rate of convergence. In Figure 11 we display the convergence history of the two-grid solver when the number of smoothing steps is varied; CG is the smoother considered for those experiments. We observe that using a too large value of  $\xi$  might delay the convergence. These situations occur when the smoother has succeeded to reduce most of the high frequencies of the error and makes only few progress in reducing the low frequencies. Applying a coarse grid correction in that case is an efficient way to speed-up the convergence. Based on this observation, an heuristic could be implemented to adapt the number of smoothing steps automatically, but we do not pursue further this aspect in this work.

In Figure 12, we compare the convergence history of the INIT-CHEBYSHEV algorithm obtained with a basis  $\mathbf{W}$  computed with a level of filtering  $\varepsilon$  equal to  $10^{-16}$  versus the Chebyshev based two-grid solver with a number of smoothing steps  $\xi = 5$  and a “near”-invariant  $\mathbf{W}$  basis precomputed with a level  $\varepsilon$  of  $10^{-4}$ . As mentioned in § 4.6, when the basis  $\mathbf{W}$  is computed with an ever worse accuracy ( $\varepsilon = 10^{-4}$ ), the convergence history of INIT-CHEBYSHEV is disrupted with a stagnation of the backward error at a level close to the actual value of  $\varepsilon$ . As opposed to this, the two-grid cycle solver

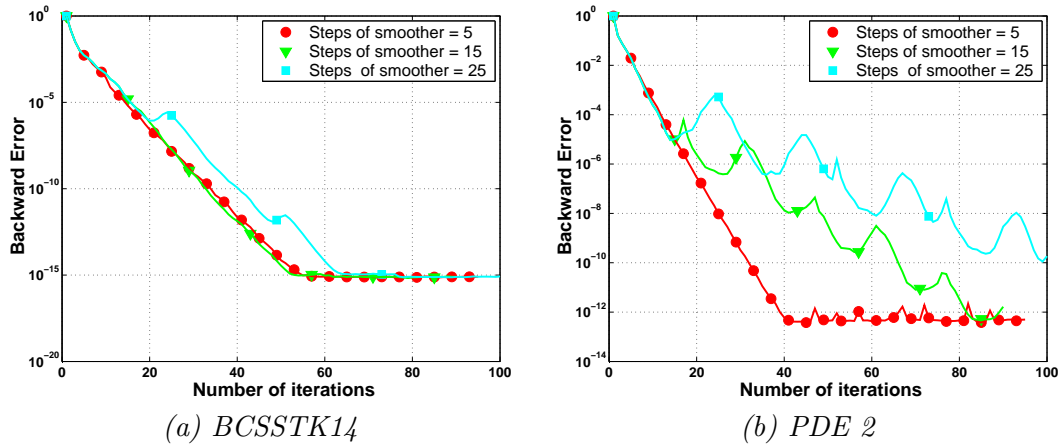


Figure 11: Convergence history of the two-grid cycle solver when the number of smoothing steps using CG is varied. The level of filtering is  $\varepsilon = 10^{-4}$

with Chebyshev as smoother using a level of filtering  $\varepsilon = 10^{-4}$  has the same curve of convergence that the INIT-CHEBYSHEV algorithm obtained with a very accurately ( $\varepsilon = 10^{-16}$ ) precomputed  $\mathbf{W}$  basis. This illustrates the benefit of using a two-grid cycle scheme when the spectral information is not very accurate.

For the test examples we have examined in our experiments we saw that the convergence rate of the two-grid cycle scheme using CG as smoother and a suitable value of  $\xi$  (i.e. not too large) has a convergence history very similar to the one obtained with SLRU when  $\varepsilon$  is small enough (i.e. smaller or equal to  $10^{-4}$ ). In that case, in terms of floating-points the two-grid cycle scheme outperforms the preconditioned CG as it performs much less oblique projections. This observation is not longer true when the “near”-invariant eigenspace poorly approximates the eigenspace (i.e.  $\varepsilon = 10^{-2}$  in our

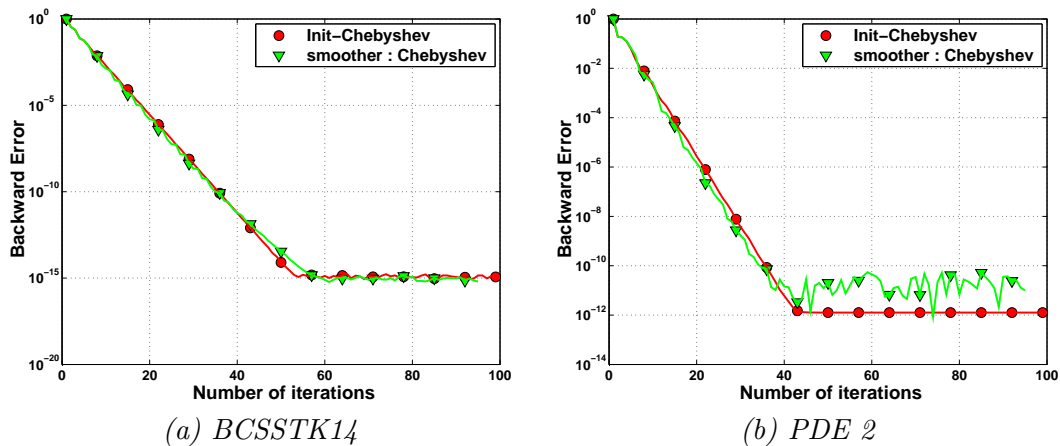


Figure 12: Convergence history of the INIT-CHEBYSHEV algorithm obtained with a basis  $\mathbf{W}$  computed with a level of filtering  $\varepsilon$  equal to  $10^{-16}$  versus the Chebyshev based two-grid solver with a number of smoothing steps  $\xi = 5$  and a “near”-invariant  $\mathbf{W}$  basis precomputed with a level  $\varepsilon$  of  $10^{-4}$

experiments). In that situation the two-grid cycle scheme does not converge anymore with any smoother as the coarse grid correction systematically corrupts the iterates computed by the smoother. We recall (see Figure 7) that SLRU and DEF-CG still succeeds to take advantage of the approximate eigenspace in that case as it was also observed in [3]. Finally we mention that the two grid cycle using a stationary method as a smoother can also be used to define a preconditioner for CG, we refer to [4] for more details on that approach.

## 5 Cost benefit and concluding remarks

In this paper, we have compared various iterative solution techniques that exploit partial spectral information about the matrix  $\mathbf{A}$ . The common general idea in all these techniques is to use some precomputed “near”-invariant subspace  $\mathbf{W}$  to build a projection onto the orthogonal complement of  $\mathbf{W}$  in some appropriate norm, and to exploit such a projector to maintain, implicitly or explicitly all through the iterations, the computed residuals in this orthogonal complement.

We have considered, for instance, the deflated version of the conjugate gradient [25] algorithm, which consists in augmenting the generated Krylov space with the basis  $\mathbf{W}$ , and resumes in projecting at each conjugate gradient iteration the generated Krylov vectors. An alternative technique consists in applying the conjugate gradient algorithm to a projected system [17]. A third approach is based on a spectral low-rank update preconditioning technique [3], which aims to shift close to one the smallest eigenvalues in the original system. These three Krylov based techniques enabled faster convergence in the conjugate gradient, with very similar linear rates of convergence that could be achieved even when the pre-computed “near”-invariant basis  $\mathbf{W}$  was obtained with very low accuracy.

For sake of comparison, we have also experienced with three other “more simple” approaches, namely the classical conjugate gradient algorithm and the Chebyshev and Richardson polynomial iterative methods, all three combined with a starting guess obtained with an oblique projection of the initial residual onto the orthogonal complement of  $\mathbf{W}$ . We have seen that, when the basis  $\mathbf{W}$  is computed with an accuracy close to machine precision, these three other techniques also present similar numerical behavior, with linear rates of convergence that differ slightly at any rate since their intrinsic numerical properties are different.

The resulting linear rates of convergence observed in all cases are directly connected to the cut-off value  $\mu$ , which corresponds to the smallest eigenvalue in  $\mathbf{A}$  that is not included in the set of eigenvalues approximated by the basis  $\mathbf{W}$ . More precisely, these rates correspond to those one would obtain with matrices having a reduced condition number equal to  $\lambda_{max}(\mathbf{A})/\mu$ . In that respect, these last three techniques are very appealing since they exploit the oblique projection with the basis  $\mathbf{W}$  only at the beginning. Finally, despite their relatively higher linear rates of convergence, the Chebyshev and Richardson polynomial iterative techniques present the additional advantage of not using any dot-products in their iterations, which can be quite beneficial when considering parallel implementations on distributed memory multiprocessor environments.

In summary, when the “*near*”-invariant subspace basis  $\mathbf{W}$  is computed with very high accuracy (i.e. with a filtering level  $\varepsilon$  close to machine precision), the coarse-grid correction need to be applied only once and algorithms of the type INIT-CG and INIT-CHEBYSHEV are the algorithms of choice, because they avoid the use of oblique projectors at each iteration and are much less expensive. When the computed “*near*”-invariant subspace basis  $\mathbf{W}$  is not very accurate, the simple use of a projector only at the beginning is not sufficient to ensure a linear rate of convergence all through the iterations. In particular, when the “*near*”-invariance of the precomputed basis is very poor, no improvements might be observed. As opposed to that, the first three Krylov techniques, which perform at each iteration an oblique projection with the  $\mathbf{W}$  basis, do not suffer from these mis-convergence problems and show a constant numerical behavior in almost every cases. In the intermediate situations, e.g. when the precomputed “*near*”-invariant basis is not extremely but sufficiently accurate, we can use a two-grid cycle scheme to still benefit from the spectral information without the expense of performing a coarse grid correction (oblique projection) at each iteration.

Concerning the pre-computation of the “*near*”-invariant basis  $\mathbf{W}$ , we have used the partial spectral factorization algorithm proposed by [2], which involves the use of Chebyshev polynomials as a filtering tool within the block Lanczos algorithm. This eigencomputation has a cost that depends on the dimension and on the accuracy of the computed “*near*”-invariant subspace and, to be effective, the gains obtained in the acceleration of the convergence of the given iterative solvers must compensate in some way the extra cost for the computation of this spectral information.

In particular, if one is interested in solving consecutively several linear systems with the same coefficient matrix but different right-hand sides, then the gains obtained can reduce quite substantially the total computational time in the long run. Finally, we illustrate this last issue experimentally in the case of the SLRU preconditioning technique. In Table 4, we indicate the minimum number of consecutive conjugate gradient runs (preconditioned by means of the SLRU preconditioning technique) before the gains obtained actually counterbalance the price for the preliminary computation of the  $\mathbf{W}$  basis with some given filtering level  $\varepsilon$ . The cost, evaluated in terms of total number matrix-vector products, of this preliminary partial spectral factorization are detailed in Table 3. We can observe that this extra cost can indeed be overcome in very few consecutive runs of the iterative solver, depending obviously on the desired level of accuracy when precomputing the invariant subspace, and also provided that this accuracy is not too low in which case the SLRU preconditioner might not be so effective, as we can observe for instance when the filtering level is only set to  $10^{-1}$ .

## Acknowledgement

We would like to thank Serge Gratton for the fruitful discussions during the development of this work. We are grateful to Mario Arioli for his encouragements and for his constructive comments on an earlier version of this paper.

Level ( $\varepsilon$ )	Number of Amortization Vectors						
	BCSSTK27	BCSSTK14	BCSSTK15	BCSSTK16	S1RMQ4M1	PDE 1	PDE 2
$10^{-16}$	10	18	8	21	21	14	4
$10^{-8}$	7	14	6	13	13	10	2
$10^{-4}$	6	11	4	10	7	10	1
$10^{-3}$	5	9	4	8	6	8	1
$10^{-2}$	4	7	3	6	4	6	2
$10^{-1}$	3	4	2	19	4	13	3

Table 4: *Number of amortization vectors. The computation of the amortization vectors is relative to SLRU and a tolerance of  $10^{-10}$ .*

## References

- [1] M. ARIOLI AND D. RUIZ, *Block conjugate gradient with subspace iteration for solving linear systems*, in Iterative Methods in Linear Algebra, II. Volume 3 in the IMACS Series in Computational and Applied Mathematics. Proceedings of The Second IMACS International Symposium on Iterative Methods in Linear Algebra., S. D. Margenov and P. S. Vassilevski, eds., 1995.
- [2] ———, *A Chebyshev-based two-stage iterative method as an alternative to the direct solution of linear systems*, Technical Report RAL-TR-2002-021, Rutherford Appleton Laboratory, Atlas Center, Didcot, Oxfordshire, OX11 0QX, England, 2002.
- [3] B. CARPENTIERI, I. DUFF, AND L. GIRAUD, *A class of spectral two-level preconditioners*, SIAM Journal on Scientific Computing, 25 (2003), pp. 749–765.
- [4] B. CARPENTIERI, L. GIRAUD, AND S. GRATTON, *Additive and multiplicative spectral two-level preconditioners for general linear systems*, Technical Report TR/PA/04/38, CERFACS, Toulouse, France, 2004.
- [5] P. CONCUS, G. GOLUB, AND D. O’LEARY, *A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations*, Technical Report STAN-CS-76-533, Stanford University, 1976.
- [6] H. DE GERSEM AND K. HAMEYER, *A deflated iterative solver for magnetostatic finite element models with large differences in permeability*, Eur. Phys. J. Appl. Phys., 13 (2000), pp. 45–49.
- [7] I. DUFF, R. GRIMES, AND J. LEWIS, *Sparse matrix test problems*, ACM Transactions on Mathematical Software, 15 (1989), pp. 1–14.
- [8] G. GOLUB AND M. KENT, *Estimates of eigenvalues for iterative methods*, Mathematics of Computation, 53 (1989), pp. 249–263.

- [9] G. GOLUB AND R. UNDERWOOD, *The Block Lanczos method for computing eigenvalues*, in Mathematical Software Symposium, University of Wisconsin, Madison, 1977. Mathematical Software III: Proceedings of a Symposium Conducted by the Mathematics Research Center, the University of Wisconsin, Madison, March 28–30, 1977, J.R. Rice, ed., no.39 in Publication of the Mathematics Research Center, the University of Wisconsin, Madison, Academic Press, New York, pp. 361–377, 1977.
- [10] G. GOLUB AND C. VAN LOAN, *Matrix Computations*, Johns Hopkins Studies in the Mathematical Sciences. The Johns Hopkins University Press, Baltimore, MD, USA, third edition, 1996.
- [11] W. HACKBUSCH, *Multigrid methods and applications.*, springer, 1985.
- [12] L. HAGEMAN AND D. YOUNG, *Applied Iterative Methods*, Academic Press, New York and London, 1981.
- [13] M. HESTENES AND E. STIEFEL, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Standards, 49 (1952), pp. 409–435.
- [14] E. KAASSCHIETER, *Preconditioned conjugate gradient for solving singular systems*, Journal of Computational and Applied Mathematics, 24 (1988), pp. 265–275.
- [15] L. KOLOTILINA, *Twofold deflation preconditioning of linear algebraic systems*, Technical Report EM-RR 20/95, Elegant Mathematics, Inc, 1995. Available at: <http://www.elegant-math.com/abs-emrr.htm>.
- [16] R. LEHOUCQ, D. SORENSEN, AND C. YANG, *ARPACK User's guide: Solution of large-scale problem with implicitly restarted Arnoldi methods*, SIAM, Philadelphia, 1998.
- [17] R. NABBEN AND C. VUIK, *A comparison of deflation and coarse grid correction applied to porous media flow*, Report 03-10, Delft University of Technology, Department of Applied Mathematical Analysis, Delft, 2003.
- [18] R. NICOLAIDES, *Deflation of conjugate gradients with applications to boundary value problems*, SIAM Journal on Numerical Analysis, 24 (1987), pp. 355–365.
- [19] D. O'LEARY, *A generalized conjugate gradient algorithm for solving class of quadratic programming problems*, Linear Algebra and its Applications, 34 (1980), pp. 371–399.
- [20] C. PAIGE, *Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem*, Linear Algebra and its Applications, 34 (1980), pp. 235–258.
- [21] B. PARLETT, *A new look at the Lanczos algorithm for solving symmetric systems of linear equations*, Linear Algebra and its Applications, 29 (1980), pp. 323–346.
- [22] Y. SAAD, *On the rates of convergence of the Lanczos and the block Lanczos methods*, SIAM Journal on Numerical Analysis, 17 (1980), pp. 687–706.

- [23] Y. SAAD, *On the Lanczos method for solving symmetric linear systems with several right-hand sides*, Mathematics of computations, 178 (1987), pp. 651–662.
- [24] ———, *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, Boston, 1996.
- [25] Y. SAAD, M. YEUNG, J. ERHEL, AND F. GUYOMARC'H, *A deflated version of the conjugate gradient algorithm*, SIAM Journal on Scientific Computing, 21 (2000), pp. 1909–1926.
- [26] R. UNDERWOOD, *An iterative block Lanczos method for the solution of large sparse symmetric eigenproblems*, PhD Thesis STAN-CS-75-496, Computer Science Department, Stanford, California, 1975.
- [27] A. VAN DER SLUIS AND H. VAN DER VORST, *The rate of convergence of conjugate gradients*, Numerische Mathematik, 48 (1986), pp. 543–560.
- [28] H. VAN DER VORST, *An iterative method for solving  $f(A)x = b$  using Krylov subspace information obtained for the symmetric positive definite*, Journal Computational and Applied Mathematics, 18 (1987), pp. 249–263.
- [29] F. VERMOLEN AND C. VUIK, *The influence of deflation vectors at interfaces on the deflated conjugate gradient method*, Report 01-13, Delft University of Technology, Department of Applied Mathematical Analysis, Delft, 2001.